ESCUELA SUPERIOR DE INGENIERÍA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

Informática Training

Jacinto Capote Robles

16 de mayo de 2010

ESCUELA SUPERIOR DE INGENIERÍA

INGENIERO TÉCNICO EN INFORMÁTICA DE SISTEMAS

Informática Training

- Departamento: Lenguajes y sistemas informáticos
- Directores del proyecto: Manuel Palomo Duarte y Antonio Balderas Alberico
- Autor del proyecto: Jacinto Capote Robles

Cádiz, 16 de mayo de 2010

Fdo: Jacinto Capote Robles

Agradecimientos

Me gustaria agradecer y dedicar este texto a mis tutores de proyecto, a mi madre por preguntarse que hacía tantos días delante del ordenador y Ignacio Palomo Duarte por sus ayudas en los diseños de interfaces.

Licencia

Índice general

Indice de figuras

Capítulo 1

Motivación y contexto del proyecto

1.1. Introducción

Aprender de una forma amena y sencilla es algo que siempre intenta buscar una persona a la hora de estudiar. El estudio de una materia, oposición ,... es más fácil si lo realizamos a través de preguntas. Además, tener registrado nuestro progreso es una forma de ir autoevaluando el nivel que vamos adquiriendo.
Un ejemplo de esta nueva forma de aprendizaje la podemos observar en la metodología que están llevando a cabo las autoescuelas en las cuales a bases de unos test y su evolución podemos aprender las normas de circulación de una forma sencilla y agradable para las personas.
Esta aplicación nace de la idea de poder enseñar a los alumnos nuevos de la carrera de informática principios básicos sobre temas informáticos que son muy importantes a lo largo de la carrera y que deberían ya de conocer.
Además la aplicación no se queda ahí si no que intenta expandirse a otros sectores como puede ser la preparación de una oposición o como en el ejemplo anterior prepararse para el carnet de conducir. El por qué de realizar esta aplicación procede de las pocas aplicaciones que existen de software libre para la realización de test. Además de que los existentes tiene una interfaz poco amena y usable esto hace que el usuario no utilize estos métodos.

1.2. Objetivos

Los objetivos de la aplicación son que el usuario que la vaya a utilizar sea capaz de realizar un estudio eficaz y controlado sobre varias materias concretas. Con ello podrá ahorrar tiempo para dedicar a otras tareas. Con la realización de estas actividades pretendo que los usuarios sean más organizados y planificadores sobre la preparación de los temas a estudiar. Estos objetivos son generales luego en el análisis de la aplicación entraremos más a fondo y de una forma más formal.

1.3. Organización del documento

El documento se divido en las siguientes secciones:

- **Introducción y motivación:** Se explicará de forma breve sobre que trata el proyecto además de la motivación que ha llevado a realizarlo.
- **Conceptos iniciales:** Se explicará conceptos claves para entender el proyecto.
- **Calendario:** Organización y planificación temporal a través de un diagrama de Ganntt.

- **Desarrollo:** Análisis, diseño e implementación de la aplicación.
- **Pruebas:** Pruebas realizadas a la aplicación para ver que tiene un funcionamiento correcto y que cumple las expectativas.
- **Conclusiones:** Conclusiones a las que he llegado tras el desarrollo de la aplicación.
- **Manuales:** Diferentes manuales para el uso de la aplicación (usuario, instalación, desarrollador y introducción a Qt).

Capítulo 2

Descripción general de la aplicación

2.1. Propósito

El propósito de la aplicación es poder realizar test de distintas materias. Para poder tener mayor amplitud estas materias a su vez se pueden dividir en categorías. Así podremos tener agrupados varios tes por categorías y materías. Los test tienen la finalidad de ayudar al estudio. Los test se podrán repetir tantas veces como se quiera así como repetir test que se han fallado muchas preguntas o test en los que hemos tenido poca puntuación.
Por lo tanto se espera que la aplicación puede ser utilizada por personas que se dedican a la docencia así como personas que desean aprender sobre un determinado tema o prepararse una oposición.

2.2. Características de usuario

La características de usuario a los que va enfocado la aplicación son personas con pocos conocimientos de informáticas (Aunque para la instalación hay que conocer un poco el sistema unix y la instalación de aplicaciones).

Capítulo 3

Conceptos iniciales

Comenzaré explicando los conceptos iniciales que se deben tener en cuenta para hacer un buen uso de la aplicación. Para que le demos un buen funcionamiento a la aplicación para aprender sobre una aplicación en contreto deberemos tener algunos aspectos en cuenta.

3.1. Enfoque de la materia a estudiar

Toda materia a estudiar con Informática Training tendrá unos pasos previos para que sea eficaz la realización de los test.
Para ser mas ilustrativos utilizaré como ejemplo para el enfoque a estudiar la materia de educación vial para la obtención del carnet B:

1. Se deberá recabar toda la información a cerca de la educación vial para la obtención del carnet B.
2. Una vez recopilado todo deberemos separar en distintas categorías la materia (Normas generales de comportamiento, señales de tráfico,...).
3. Una obtenidas las categorías es muy importante obtener los aspecto principales de la categorización realizada. Por ejemplo para las señales de tráfico podría ser (tipos de señales, señales de prohibición, ...).
4. Una vez realizado el anterior paso deberemos realizar preguntas que identifiquen adecuadamente estos aspectos principales sin dejarnos nada importante. Las preguntas podremos realizarlas de verdadero falso, basándonos en una ilustración, 4 opciones ,...
5. Por último solo deberemos introducirlas en Informática Training para poder empezar a realizar diferentes test acerca de la materia creada. En este caso educación vial para la obtención del carnet B.

3.2. Metodología de estudio

El pensamiento que se ha tenido a la hora de desarrollar la aplicación es la aplicación de distintas metodologías de estudios. Para que sirviera de ejemplo se ha definido la siguiente metodología de estudio. Siguiendo el ejemplo de antes el usuario desea comenzar el estudio por lo que primero deberá realizar una lectura a fondo sobre la materia. Una vez realizada la lectura procederá a realizar el test completo de una categoría por ejemplo señales de tráfico. Una vez finalizado puede comprobar si ha asimilado adecuadamente el tema o no. En caso de que no lo tenga asimiliado puede realizar las preguntas que

haya fallado, realizar el último test,... En caso de que quiera repasar puede realizar el test completo pero con las preguntas aleatorias para que no resulten familiares las preguntas, ... Como se puede observar se pueden realizar muchas metolodogías de estudio diferentes con distinto nivel de eficiencia, tiempo, aprendizaje.

Capítulo 4

Calendario

4.1. Introducción

La distribución temporal seguida en líneas generales en la descrita a continuación:

- **Análisis:** En esta fase llevaremos a cabo la recogida de todos los requisitos de aplicación. Al seguir un modelo lineal deberemos realizar una buena recogida de requisitos para que quede una aplicación consistente y cumpliendo los objetivos deseados para ella. Una vez recogidos realizaremos los casos de usos y diagramas correspondientes.
- **Diseño:** En este fase comenzaremos realizando el diseño de las interfaces gráficas. Una vez realizadas realizaremos un estudio de usabilidad sobre ellas para que sea usable de cara al usuario.
- **Programación:** Primero realizaremos un estudio de las aplicaciones más adecuada para el desarrollo de la aplicación asi como el lenguaje de programación y la plataforma. Y la parte más importante de esta fase la codificación de aplicación.
- **Pruebas:** Las pruebas que realizaremos serán las siguientes:
  1. Test usabilidad.
  2. Pruebas caja negra.
  3. Estudio y resolución de los resultados de las pruebas.
- **Documentación:** Para la documentación del código utilizaremos doxygen para luego poder generar un documento que ayudará a futuros desarrolladores. En esta fase también realizaremos la memoria de la aplicación, manuel de usuario y la presentación para difundir la aplicación.

4.2. Diagrama de Grantt

Podemos ver en la siguiente figura 4.1 el diagrama de Gantt con los plazos de las fases de la aplicación.

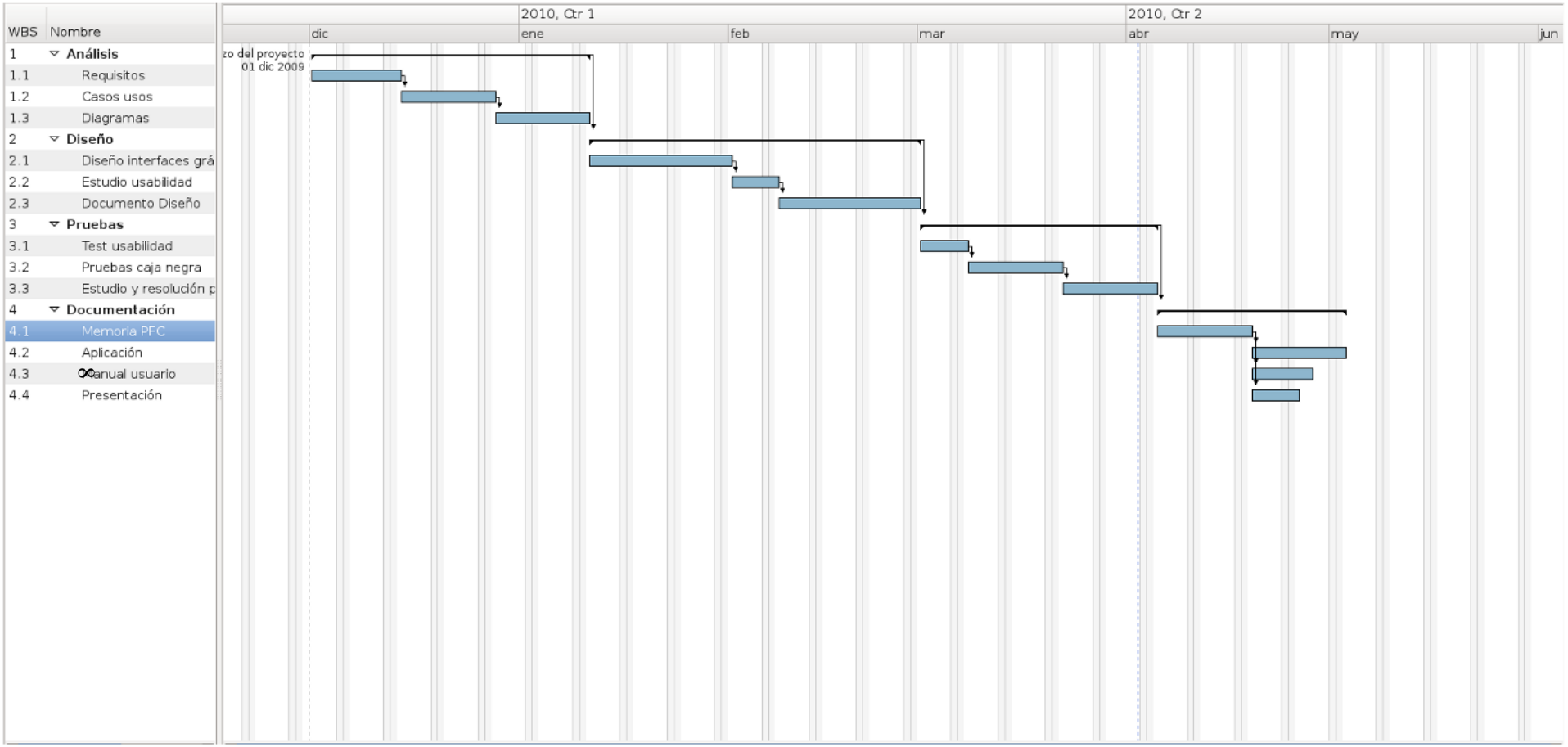

Figura 4.1: Diagrama de Gantt

Capítulo 5

Análisis

5.1. Metodología de desarrollo

Al desarrollarse un proyecto de pequeñas dimensionas y con una sola persona es un poco absurdo hablar de una metodología de desarrollo a seguir. Pero teniendo encuenta que es un proyecto de software libre y que en un futuro tendrá una comunidad que desarrolle nuevas versiones del sistema he elegido desde el principio seguir la metodología de desarrollo **eXtreme Programming**. La metodología será llevada a cabo dentro de los límites que una única persona pueda realizar.
¿En que consiste la metodología de desarrollo eXtreme Programing?
Es un enfoque de la ingeniería del software formulado por **Kent Beck**. Es el más destacado de los procesos ágiles de desarrollo software. Sus características principales son:

- Desarrollo iterativo e incremental.
- Pruebas unitarias continuas frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión.
- Programación en parejas.
- Correción de todos los errores antes de añadir nueva funcionalidad.
- Refactorización del código.
- Propiedad del código compartida: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto.
- Simplicidad en el código(es la mejor forma de que funcionen las cosas).

5.2. Toma de requisitos

Tras un análisis profundo de que debe realizar y el alcance de la aplicación los requisitos que he obtenido son:

- Realizar test de preguntas.
- Los test se podrán exportar a PDF.
- Los test se podrán configurar (tiempo, número preguntas, matería, categoría, preguntas erróneas,..)

- Las preguntas de los test pueden llevar imagen asociada.
- Las preguntas pueden ser de verdadero o falso (dos opciones) o de 4 opciones.
- Se pueden crear preguntas nuevas.
- Las preguntas se asociación mediante una materia y una categoría.
- Se pueden crear materías y categorías.
- Las preguntas, materias y categorías se deben poder importar desde xml.
- Se podrán dar de alta y baja a usuarios.
- Se podrá obtener estadísticas de usuarios sobre los test realizados.
- Las estadísticas serán en torno a las categorías y las materias.
- Playlist de canciones.
- Al finalizar los test se podrán observar las preguntas falladas así como la puntuación obtenida.
- Multidioma (inglés, español, italiano)
- Únicamente existirá un rol de usuario.

5.3. Diagrama de casos de usos

En la siguiente imagen 5.1 podemos observar el diagrama de casos de uso asociada a la toma de requisitos.

5.4. Definición de los casos de usos

5.4.1. Caso de uso gestión categorías

Descripción: Administrar categorías.
Actores: Usuario.
Precondición: Ninguna.
Postcondición: Se da de alta una nueva categoría.
Escenario principal:

1. El usuario desea crear una nueva categoría.
2. El usuario introduce el nombre y selecciona la materia.
3. El sistema registra la categoría.

Escenario alternativo:

1a. El usuario desea editar una categoría.

1a.-1 El usuario selecciona una categoría.

1a.-2 El usuario selecciona un nuevo nombre y materia asociada.

1a.-3 El sistema comprueba de que no existe el nombre seleccionado y crea la categoría.

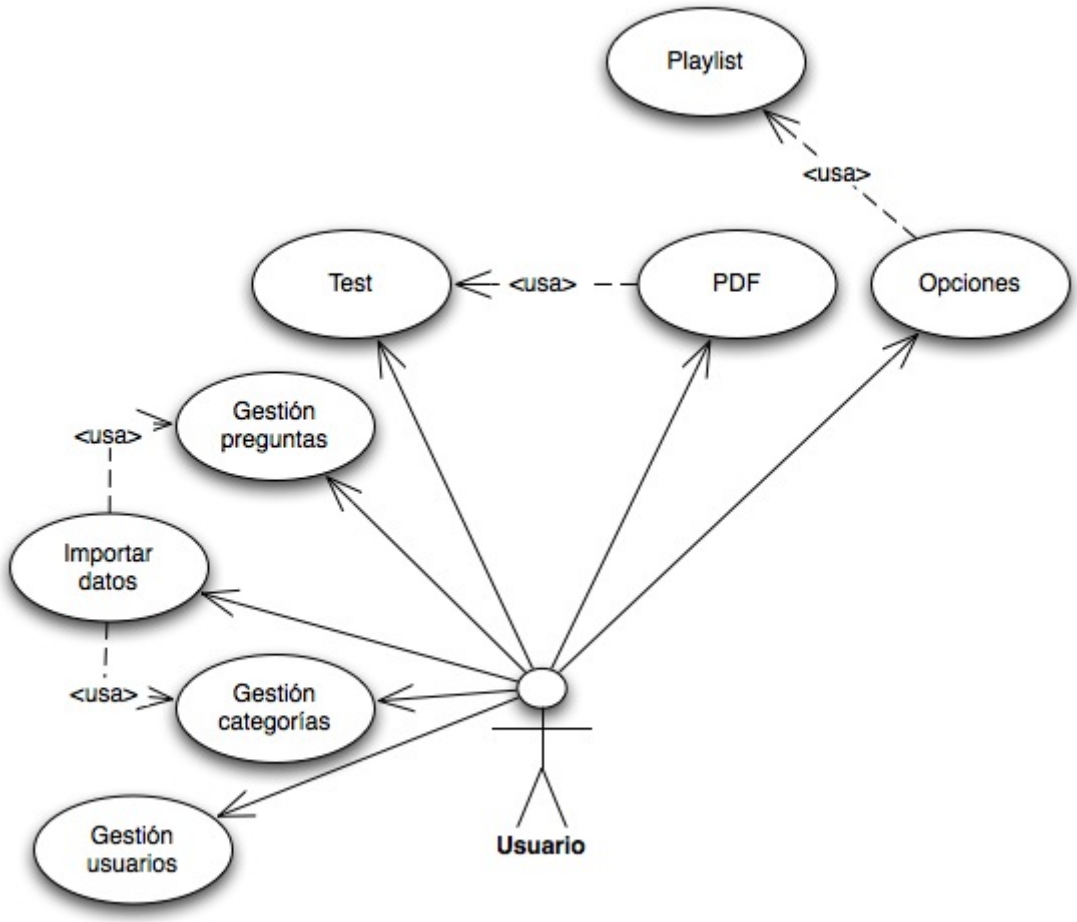

Figura 5.1: Diagrama de casos de usos

1b El usuario desea eliminar una categoría.

1b.-1 El usuario selecciona una categoría.

1b.-2 El sistema elimina la categoría y sus preguntas asociadas.

1c. El usuario desea crear una materia.

1c.-1 El usuario introduce nombre

1c.-2 El sistema comprueba de que no existe y crea el nombre

1d El usuario desea editar una materia

1d.-1 El usuario selecciona la materia

1d.-2 El usuario introduce el el nuevo nombre

1d.-3 El sistema comprueba de que no existe el nombre seleccionado y crea la materia.

1e El El usuario desea eliminar una materia.

1e.-1 El usuario selecciona una materia.

1e.-2 El sistema elimina la materia, las categorías y preguntas asociadas.

5.4.2. Caso de uso gestión usuarios

Descripción: Administrar usuarios.
Actores: Usuario.
Precondición: Ninguna.
Postcondición: Se da de alta un nuevo usuario.
Escenario principal:

1. El usuario desea registrar un nuevo usuario
2. El usuario introduce el nombre
3. El sistema comprueba de que no exista el usuario y crea el nuevo usuario

Escenario alternativo:

1a. El usuario desea editar un usuario.

1a.-1 El usuario selecciona un usuario.

1a.-2 El usuario introduce un nuevo nombre de usuario.

1a.-3 El sistema comprueba de que no exista usuario con el nombre nuevo y realiza el cambio.

1b. El usuario desea eliminar un usuario.

1b.-1 El usuario selecciona un usuario.

1b.-4 El sistema elimina el usuario.

5.4.3. Caso de uso gestión preguntas

Descripción: Administrar preguntas.
Actores: Usuario.
Precondición: Ninguna.
Postcondición: Se da de alta una nueva pregunta.
Escenario principal:

1. El usuario desea crear una nueva pregunta.
2. El usuario introduce los datos y selecciona la categoría a la que va a pertenecer.
3. El sistema valida los datos y registra la nueva pregunta.

Escenario secundario:

1a. El usuario desea editar una pregunta.

1a.-1 El usuario selecciona una pregunta.

1a.-2 El usuario cambia los datos.

1a.-3 El sistema valida los datos y guarda los cambio de la pregunta.

1b. El usuario desea eliminar una pregunta.

1b.-1 El usuario selecciona una pregunta.

1b.-2 El sistema elimina la pregunta.

5.4.4. Caso de uso importar datos

Descripción: Importa los datos a partir de un XML.
Actores: Usuario.
Precondición: XML está bien formado.
Postcondición: Se importan los datos correctamente (materias, categorías y preguntas).
Escenario principal:

1. El usuario desea importar nuevos datos.
2. El usuario selecciona un fichero XML válido.
3. El sistema valida el fichero y comienza la importación.
4. El sistema importa las materías y categorías<use>Gestión categorías.
5. El sistema importa las preguntas <use>Gestión preguntas.

Escenario alternativo:

2a. El usuario selecciona un fichero XML inválido.

2a.-1 El sistema avisa al usuario de que el fichero no es válido.

5.4.5. Caso de uso test

Descripción: Realización de un test.
Actores: Usuario.
Precondición: Ninguno.
Postcondición: Se realiza un test.
Escenario principal:

1. El usuario desea comenzar un test.
2. El usuario pulsa comenzar test.
3. El sistema genera el test obteniendo la configuración del usuario.
4. El sistema obtiene la pregunta actual.
5. El usuario responde.
6. El sistema comprueba si es la última pregunta. En caso contrario vuelve a 3.
7. El sistema muestra los resultados.

Escenario alternativo:

3a. El sistema comprueba que no hay preguntas y da el aviso.

5.4.6. Caso de uso PDF

Descripción: Obtener test en un fichero PDF.
Actores: Usuario.
Precondición: Ninguno.
Postcondición: Se obtiene un fichero PDF con un test determinado.
Escenario principal:

1. El usuario desea exportar a PDF un test.
2. El usuario pulsa exportar.
3. El sistema genera PDF obteniendo preguntas <use>Test.
4. El usuario selecciona lugar donde guardarlo.
5. El sistema guarda el fichero.

5.4.7. Caso de uso playlist

Descripción: Añade un fichero de música a la aplicación.
Actores: Usuario.
Precondición: Ninguno.
Postcondición: Se genera una playlist de canciones para utilizar en la aplicación.
Escenario principal:

1. El usuario desea generar una playlist
2. El usuario selecciona un fichero de música
3. El sistema comprueba si es válido y lo añade a la playlist.
4. El usuario decide no añadir más ficheros.

Escenario alternativo:

4a. El usuario decide añadir otro fichero más volver a 2.

5.4.8. Caso de uso opciones

Descripción: Configura la aplicación.
Actores: Usuario.
Precondición: Ninguno.
Postcondición: Se guarda la configuración de la aplicación.
Escenario principal:

1. El usuario desea cambiar la configuración.
2. El usuario selecciona una configuración determinada y configura <use>playlist.
3. El sistema valida la configuración y la guarda.

5.5. Diagrama de clases

En la siguiente imagen 5.2 podemos observar el diagrama de clase asociado a los requisitos obtenidos.

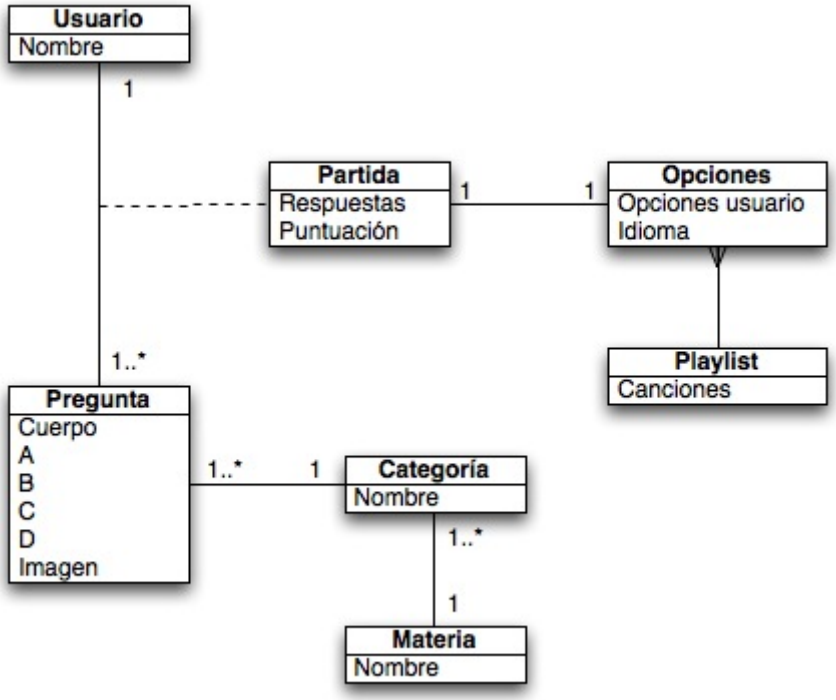

Figura 5.2: Diagrama de clases

5.6. Contrato de las operaciones

En esta sección aparecerán todos los contratos de las operaciones obtenenidas a través de los resultados anteriores.

5.6.1. Operación crear categoría

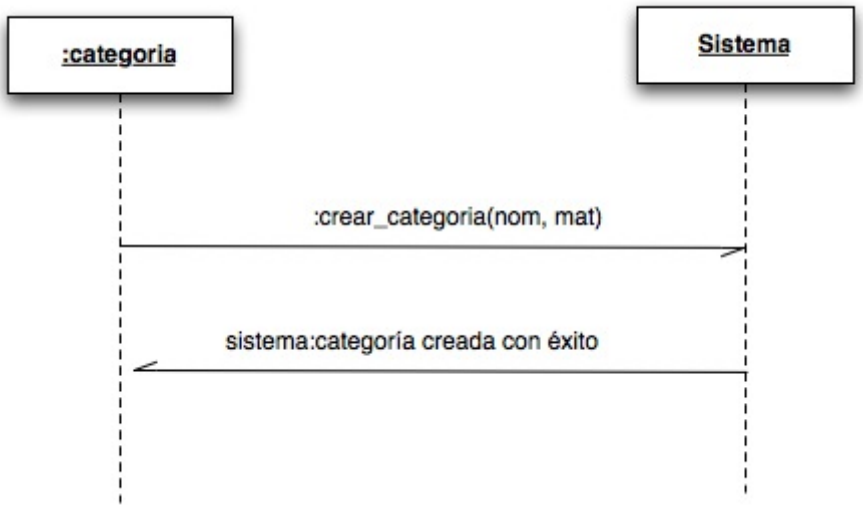

Figura 5.3: Contrato crear categoría

Operación: crear_categoria(nom, mat)
Responsabilidades: Crea una nueva categoría
Refencias cruzadas: Caso de uso: Administrar categorías
Precondición: Existe objeto materia m
Postcondición: Se crea un nuevo objecto categoria c asociada a una materia m existente y con nombre nom

5.6.2. Operación crear materia

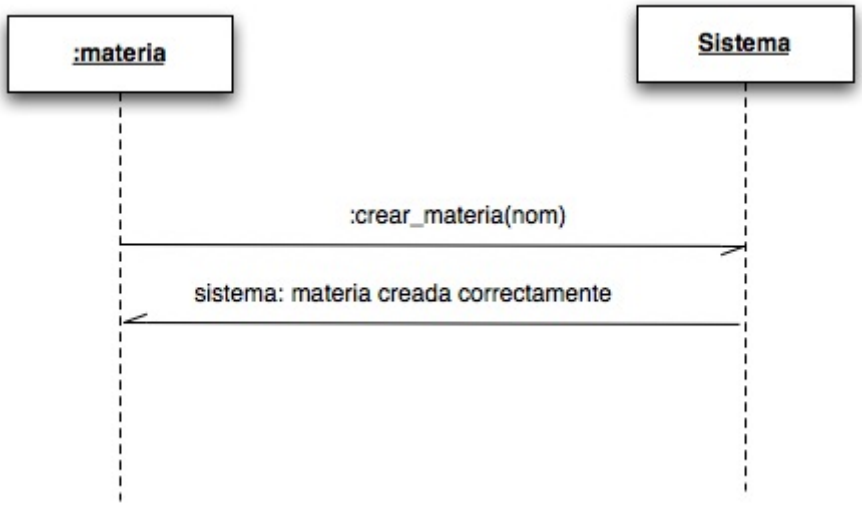

Figura 5.4: Contrato crear materia

Operación: crear_materia(nom)
Responsabilidades: Crea una nueva materia
Refencias cruzadas: Caso de uso: Administrar categorías
Precondición:
Postcondición: Se crea un nuevo objecto materia m con nombre nom

5.6.3. Operación crear pregunta

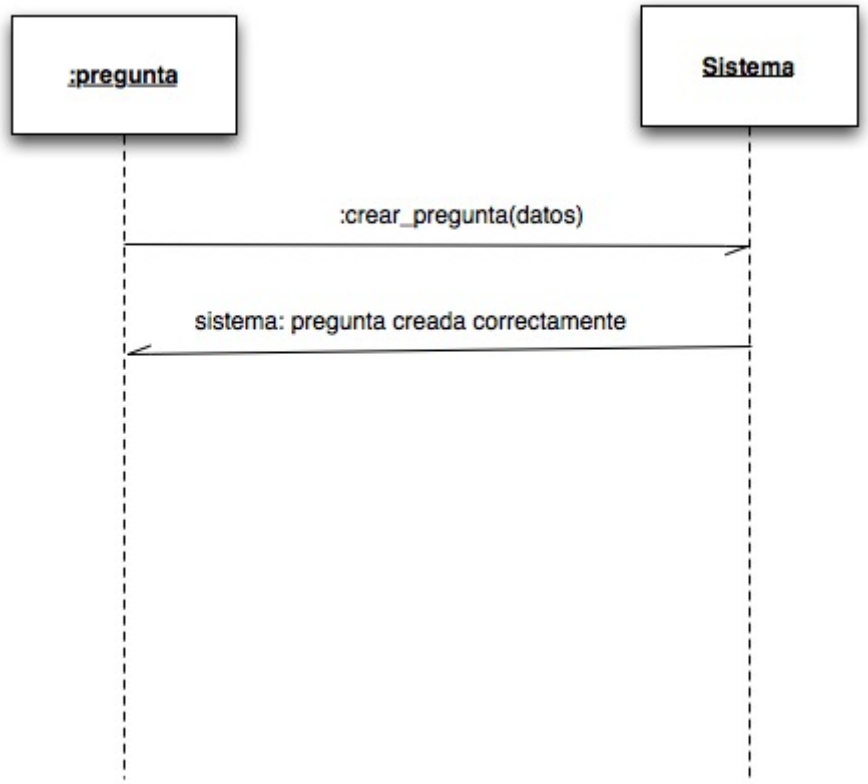

Figura 5.5: Contrato crear pregunta

Operación: crear_pregunta(datos)
Responsabilidades: Crea una nueva pregunta
Refencias cruzadas: Caso de uso: Administrar preguntas
Precondición: Existe un objecto categoria c
Postcondición: Se crea un objeto pregunta p asociando los datos con los datos del objecto. Además de obtener de los datos la asociación con el objecto c

5.6.4. Operación crear usuario

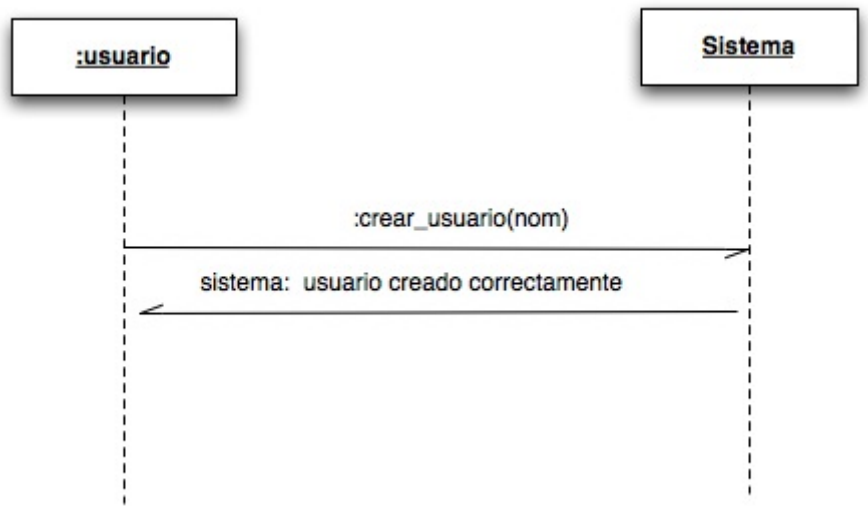

Figura 5.6: Contrato crear usuario

Operación: crear_usuario(nom)
Responsabilidades: Crea un nuevo usuario
Refencias cruzadas: Caso de uso: Administrar usuarios
Precondición:
Postcondición: Se crea un objeto usuario u con nombre nom

5.6.5. Operación editar categoría

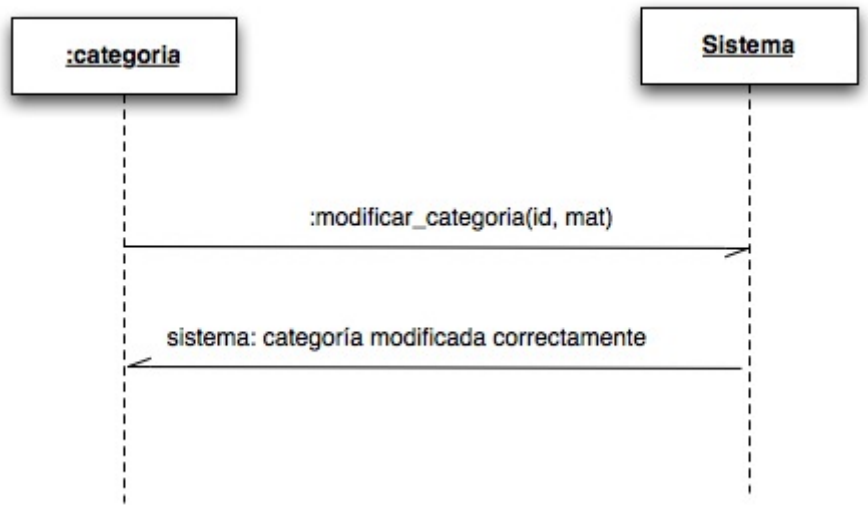

Figura 5.7: Contrato editar categoría

Operación: editar_categoria(id, datos)
Responsabilidades: Edita una categoría
Refencias cruzadas: Caso de uso: Administrar categorías
Precondición: Existe objeto materia m y un objecto categoria c
Postcondición: Se asociado el objeto c con el objecto materia m se cambia el nombre por nom

5.6.6. Operación editar materia

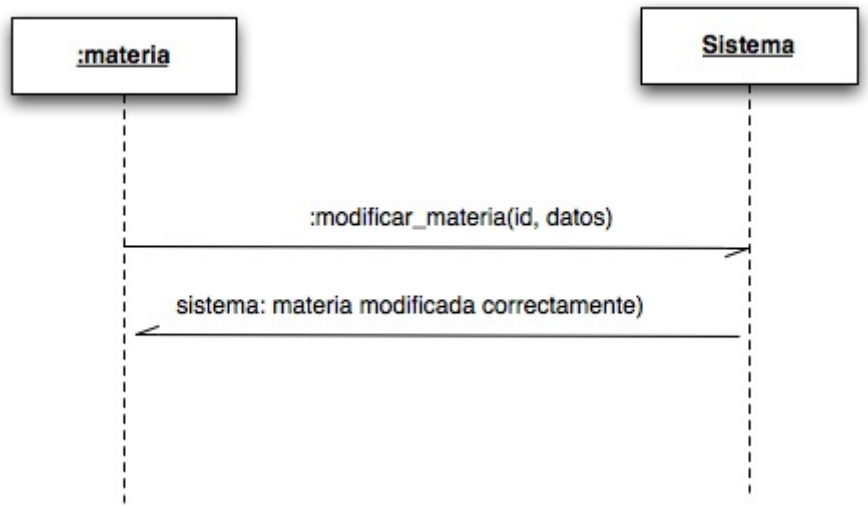

Figura 5.8: Contrato editar materia

Operación: editar_materia(id, datos)
Responsabilidades: Edita una materia
Refencias cruzadas: Caso de uso: Administrar categorías
Precondición: Existe objecto m
Postcondición: Se modifica el objeto m con nombre nom

5.6.7. Operación editar pregunta

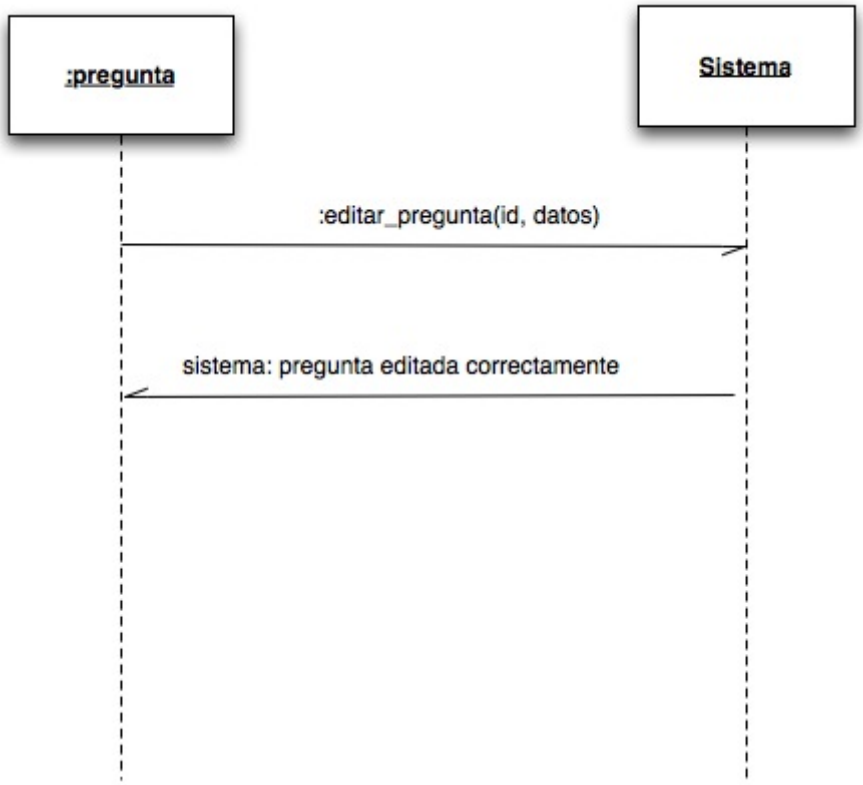

Figura 5.9: Contrato editar pregunta

Operación: editar_pregunta(id, datos)
Responsabilidades: Edita una pregunta
Refencias cruzadas: Caso de uso: Administrar preguntas
Precondición: Existe un objecto pregunta p y una categoría c
Postcondición: Se modifica el objeto p con los campos de datos. Además se modifica la asociación con la categoria c con los datos obtenidos de la variable datos.

5.6.8. Operación editar usuario

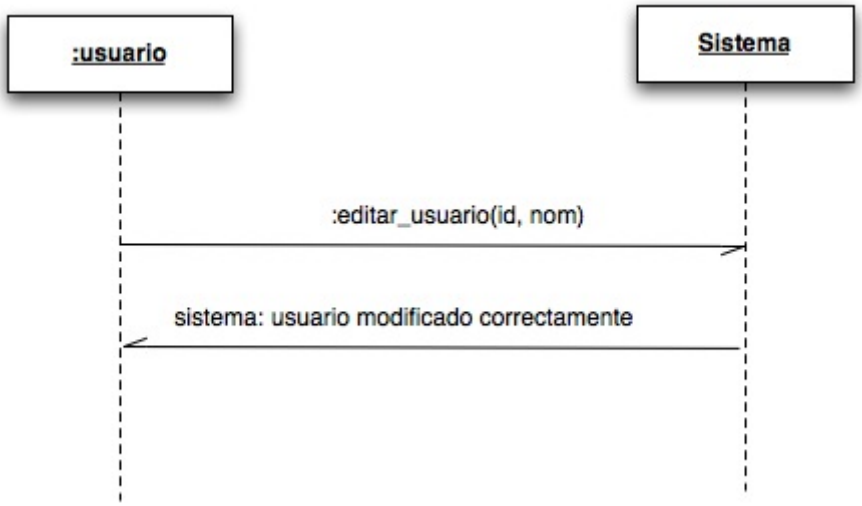

Figura 5.10: Contrato editar usuario

Operación: editar_usuario(id, nom)
Responsabilidades: Edita un usuario
Refencias cruzadas: Caso de uso: Administrar usuarios
Precondición: Existe objecto u de usuario
Postcondición: Se modifica el objecto u con nombre nom

5.6.9. Operación eliminar categoría

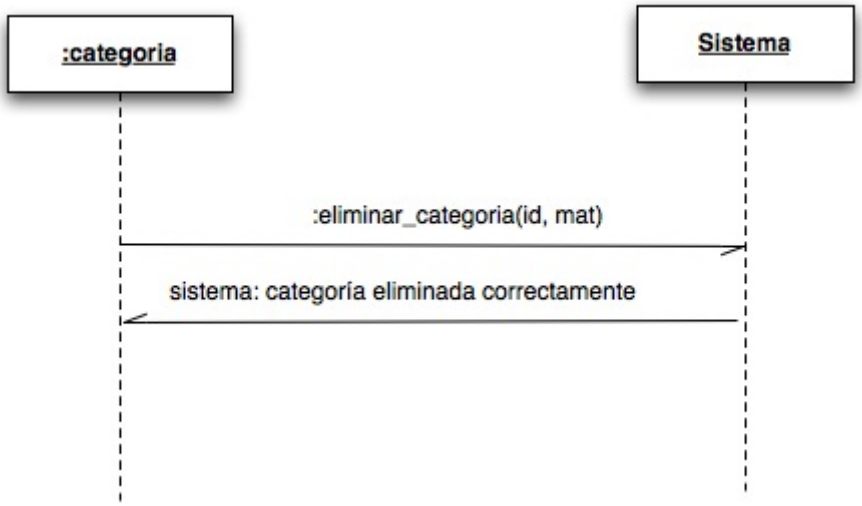

Figura 5.11: Contrato eliminar categoría

Operación: eliminar_categoria(id, mat)
Responsabilidades: Elimina una categoría
Refencias cruzadas: Caso de uso: Administrar categorías
Precondición: Existe objeto categoría c y objeto materia m
Postcondición: Se elimina la asociación con el objeto m. Se elimina el objeto c

5.6.10. Operación eliminar materia

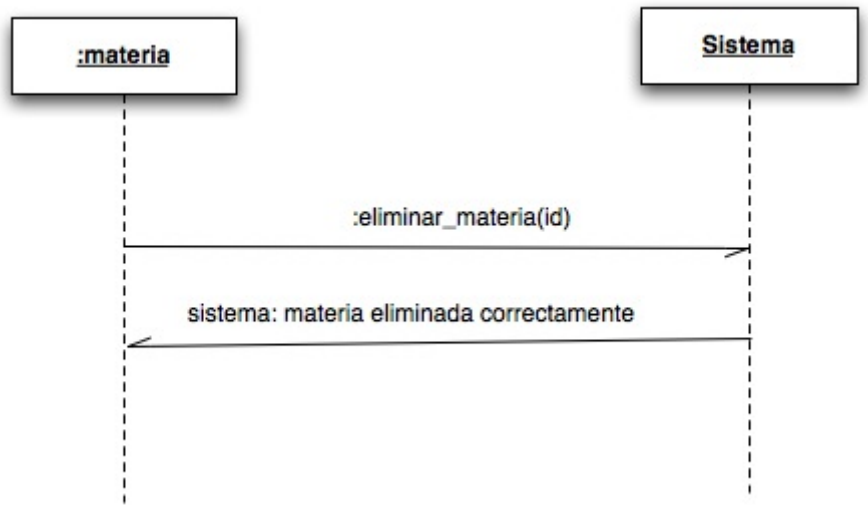

Figura 5.12: Contrato eliminar materia

Operación: eliminar_materia(id)
Responsabilidades: Elimina una materia
Refencias cruzadas: Caso de uso: Administrar categorías
Precondición: Existe objecto m
Postcondición: Se elimina objeto m

5.6.11. Operación eliminar pregunta

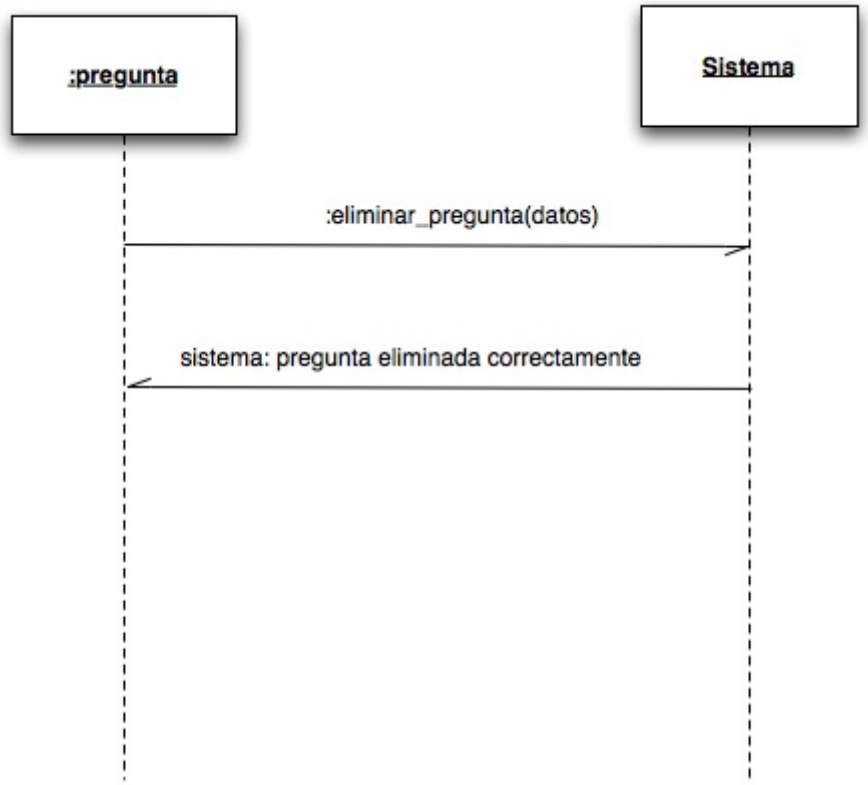

Figura 5.13: Contrato eliminar pregunta

Operación: eliminar_pregunta(datos)
Responsabilidades: Elimina una pregunta
Refencias cruzadas: Caso de uso: Administrar preguntas
Precondición: Existe un objecto pregunta p y una categoría c
Postcondición: Se elimina la relación de p con la categoria c. Luego se elimina objeto p. con la categoria c con los datos obtenidos de la variable datos.

5.6.12. Operación eliminar usuario

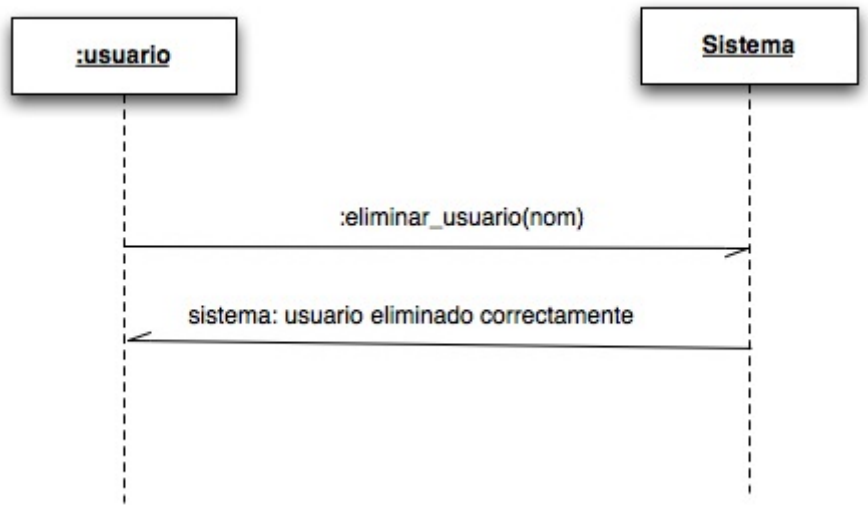

Figura 5.14: Contrato eliminar usuario

Operación: eliminar_usuario(nom)
Responsabilidades: Elimina un usuario
Refencias cruzadas: Caso de uso: Administrar usuarios
Precondición: Existe objeto u de usuario
Postcondición: Se elimina objeto u de usuario.

5.6.13. Operación importar datos

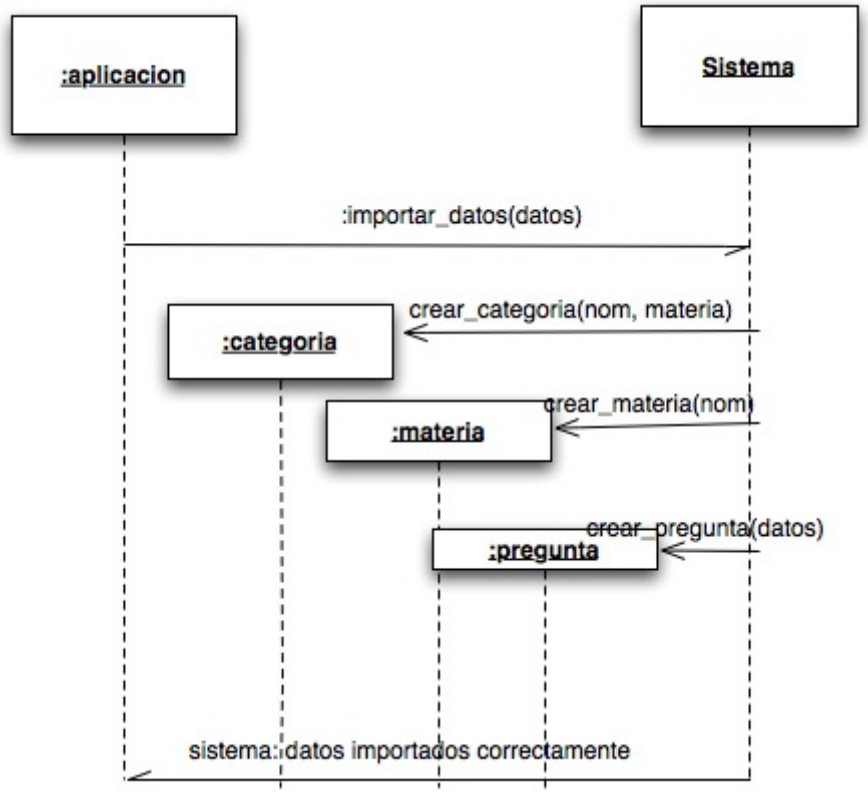

Figura 5.15: Contrato importar datos

Operación: importar_datos(datos)
Responsabilidades: Importa datos (categoría, matería, pregunta)
Refencias cruzadas: Caso de uso: Importar datos
Precondición:
Postcondición: Se van leyendo los datos y mientras no se finalice se crean las categoría, materias y preguntas que se vayan procesando.

5.6.14. Operación cambiar opciones

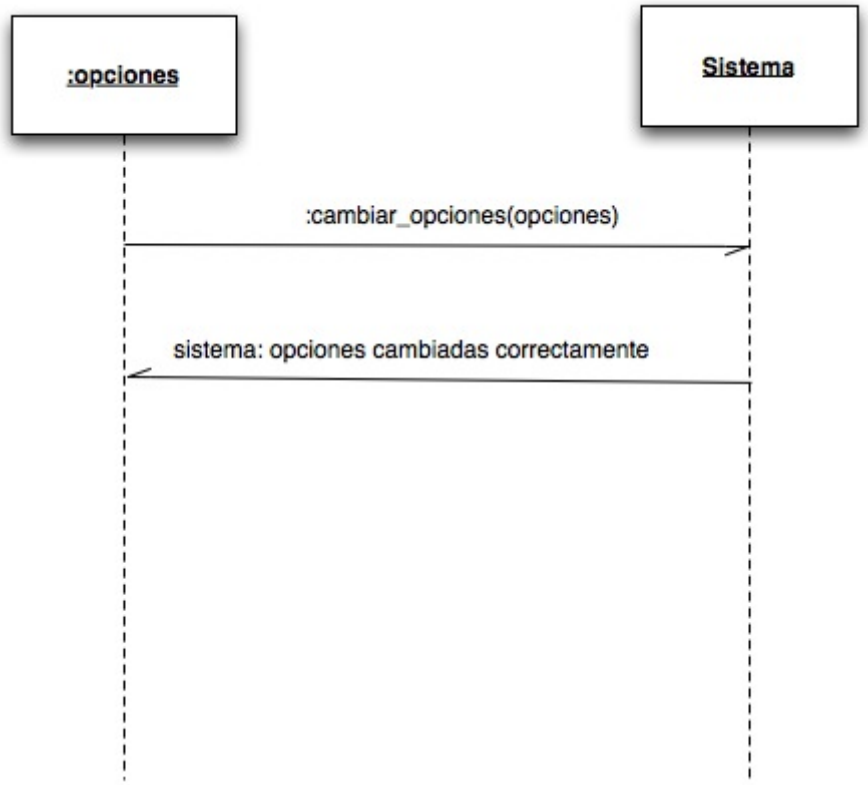

Figura 5.16: Contrato cambiar opciones

Operación: cambiar_opciones(opciones)
Responsabilidades: Cambia las opciones
Refencias cruzadas: Caso de uso: Opciones
Precondición:
Postcondición: Se crea objeto opciones o con opciones de usuario y idioma igual a variable opciones.

5.6.15. Operación generar pdf

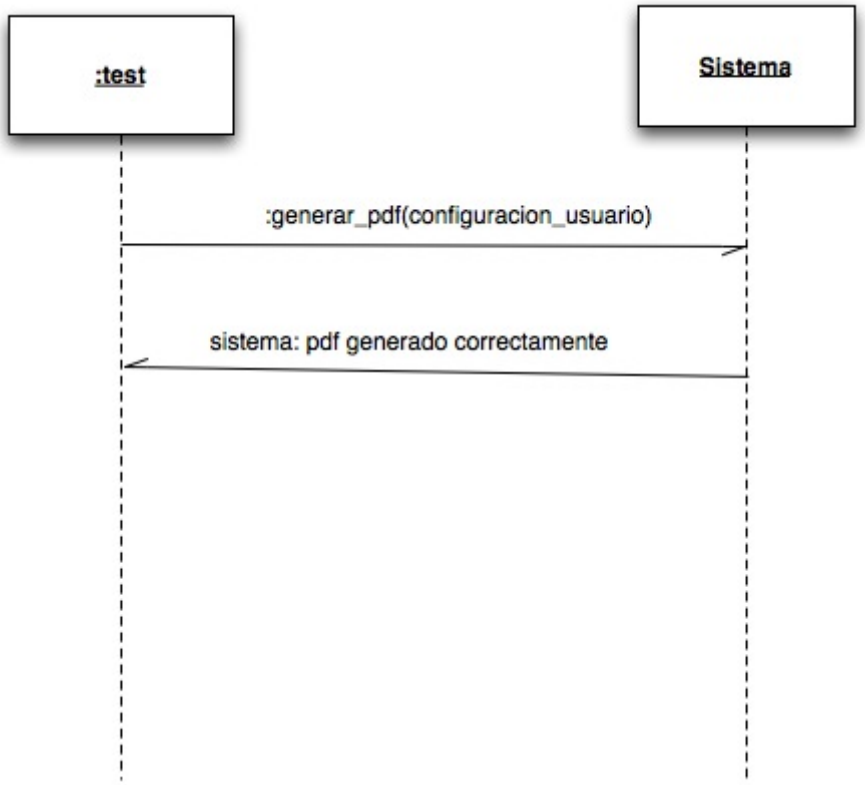

Figura 5.17: Contrato generar pdf

Operación: generar_pdf(configuracion_usuario)
Responsabilidades: Genera un documento PDF
Refencias cruzadas: Caso de uso: PDF
Precondición:
Postcondición: Se genera un pdf usuando las preguntas de la relación de usuario con pregunta obtenido a través de la configuración de usuario

5.6.16. Operación playlist

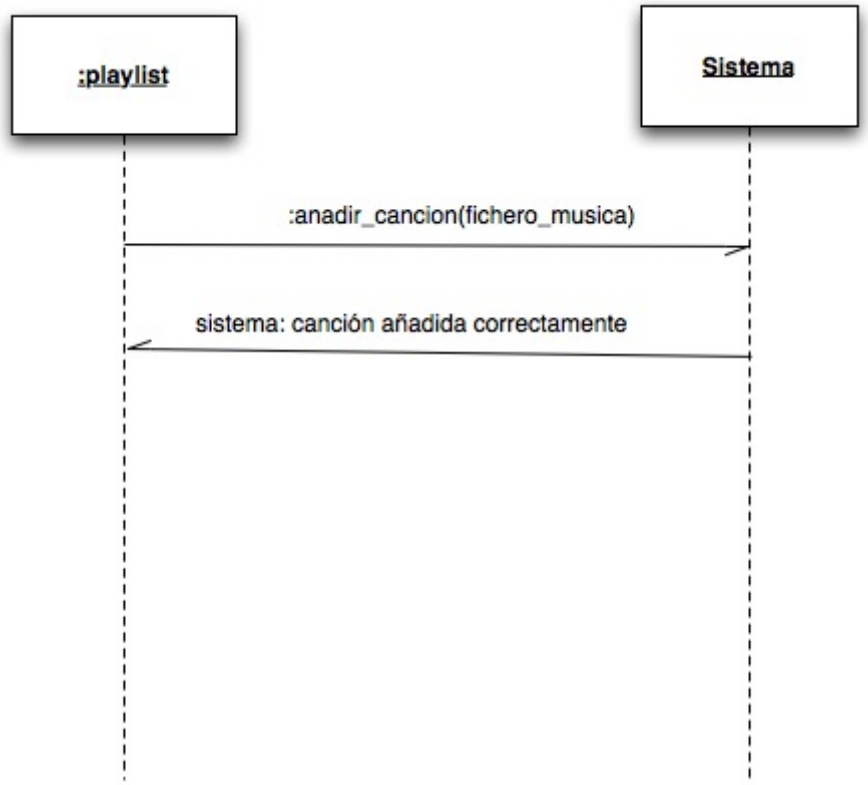

Figura 5.18: Contrato anadir canción

Operación: anadir_cancion(fichero_musica)
Responsabilidades: Añade una nueva canción a la playlist
Refencias cruzadas: Caso de uso: Playlist
Precondición: Existe un objeto p de playlist
Postcondición: Se añade fichero_musica a canciones de playlist.

5.6.17. Operación generar test

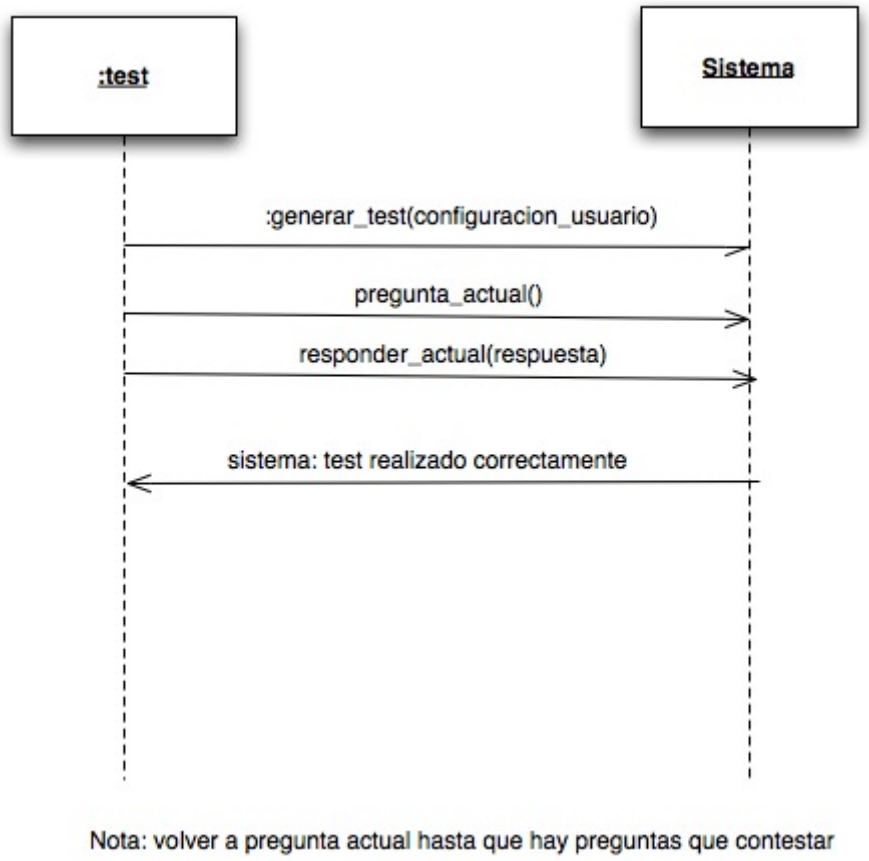

Figura 5.19: Contrato generar test

Operación: generar_test(configuracion_usuario)
Responsabilidades: Genera un test
Refencias cruzadas: Caso de uso: Test
Precondición: Existe objeto u de usuarioy varios objetos pregunta p1, p2, ...
Postcondición: Se asocia u con p1, p2, p3, ... y se crea el objeto pa partida asociado a dicha relación

5.6.18. Operación pregunta actual

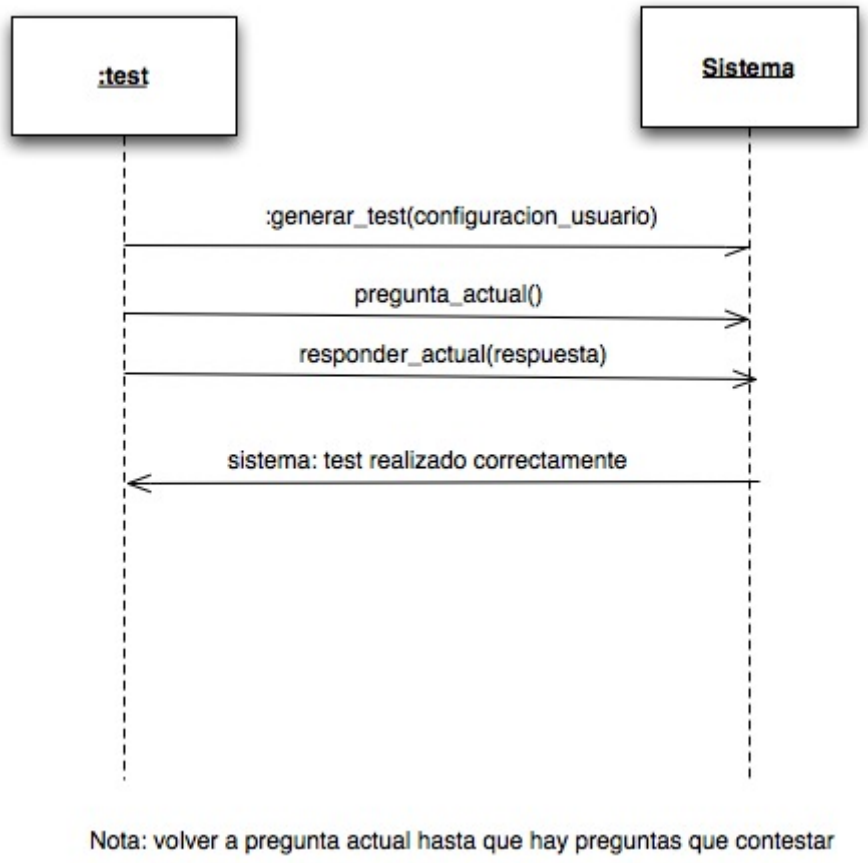

Figura 5.20: Contrato pregunta actual

Operación: pregunta_actual()
Responsabilidades:
Refencias cruzadas: Caso de uso: Test
Precondición: Existe un usuario u y pregunta p1, p2, .. con una relación creada y un objeto pa partida asociada a dicha relación.
Postcondición: Se devuelve la pregunta pn que está actualmente usándose.

5.6.19. Operación responder actual

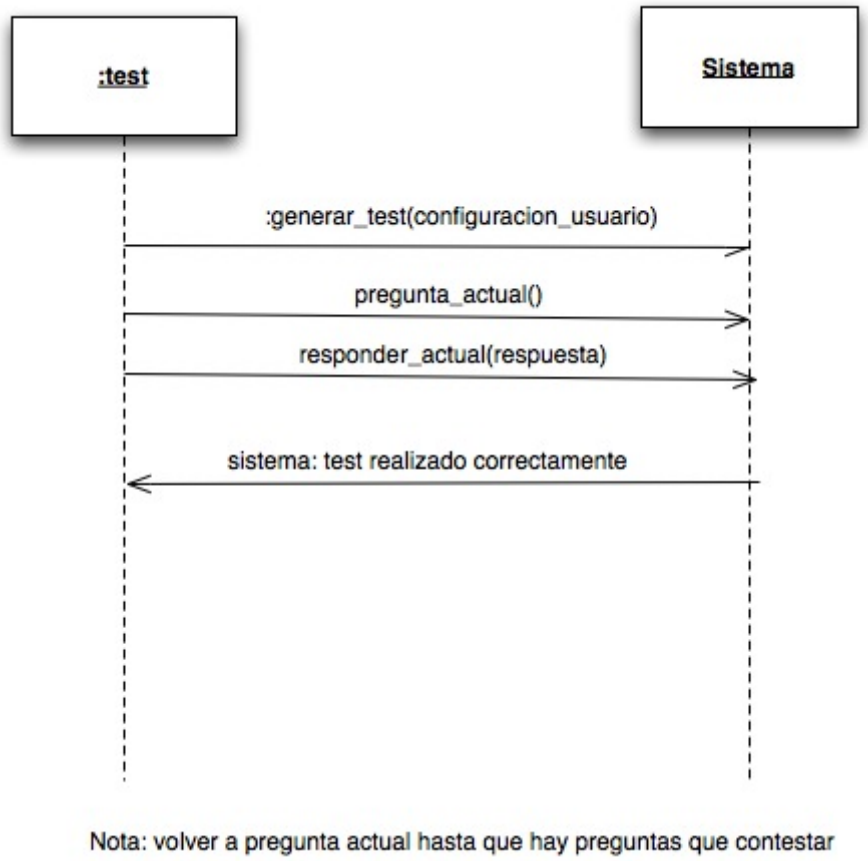

Figura 5.21: Contrato responder actual

Operación: responder_actual(respuesta)
Responsabilidades:
Refencias cruzadas: Caso de uso: Test
Precondición: Existe un usuario u y pregunta p1, p2, .. con una relación creada y un objeto pa partida asociada a dicha relación.
Postcondición: Se actualiza el atributo respuestas objeto pa partida con respuesta y se actualiza la pregunta actual de la asociación.

Capítulo 6

Diseño

6.1. Introducción

Primero comenzaré describiendo los requisitos del sistema necesarios para poder ejecutar la aplicación para posteriormente introducir las herramientas que se van a usar para el desarrollo de la aplicación.

6.2. Definición de los requisitos del sistema

Los requisitos hardware necesarios para poder ejecutar la aplicación correctamente en una computadora son los siguientes:

- Procesador superior o igual a 1,6 GHz.
- Memoria RAM mayor o igual 512 MB.
- Memoria tarjeta gráfica mayor o igual a 128 MB.

Los requisitos software necesarios son:

- Sistema unix.
- Tener aceleración 3D.
- Tener instalada las librerías para compilación de código C++ (g++).
- Tener instalada las librerías Qt-core, Qt-phonon, Qt-sqlite en su versión 4 o superior.
- Tener un sistema de sonido instalado.

6.3. Herramientas utilizadas

6.3.1. Lenguaje de programación

Como lenguaje de programación se ha obtado por C++. Dicho lenguaje nos proporciona un gran control sobre la máquina en la se ejecute la aplicación. Además es un lenguaje fácil de comprender y muy exportable para llevar la la aplicación a otros sistemas operativos que no sean unix (por ejemplo Mac OS X).
Aunque la programación sea con C++ se va a usar un capa de abstracción basada en Qt (*en el siguiente apartado se explica mejor Qt*). Con Qt conseguimos realizar un enfoque de programación orientada a eventos. Al tratar con formularios gráficos debemos tener en cuenta que se debe buscar utilidades que nos ayuden a realizar una programación orientada a eventos para usar los menos recursos posibles.

6.3.2. Diseño de interfaces

Para el diseño de interfaces he optado por usar Qt en su versión 4.x. Qt es una biblioteca multiplataforma para desarrollar interfaces gráficas de usuario y también para el desarrollo de programas sin interfaz gráfica como herramientas de la consola y servidores. Qt es utilizada principalmente en KDE, Google Earth, Skype, Qt Extended, Adobe Photoshop Album, VirtualBox y Opie. Es producido por la división de software Qt de Nokia, que entró en vigor después de la adquisición por parte de Nokia de la empresa noruega Trolltech, el productor original de Qt, el 17 de junio de 2008.
Qt es utilizada en KDE, un entorno de escritorio para sistemas como GNU/Linux o FreeBSD, entre otros. Qt utiliza el lenguaje de programación C++ de forma nativa, adicionalmente puede ser utilizado en varios otros lenguajes de programación a través de bindings. Al ser nativo con C++ obtendremos una capa de abstracción en C++ para poder implementar la aplicación.
Funciona en todas las principales plataformas, y tiene un amplio apoyo. El API de la biblioteca cuenta con métodos para acceder a bases de datos mediante SQL, así como uso de XML, gestión de hilos, soporte de red, una API multiplataforma unificada para la manipulación de archivos y una multitud de otros para el manejo de ficheros, además de estructuras de datos tradicionales.
Distribuida bajo los términos de GNU Lesser General Public License.
Resumiendo Qt se utilizará para el diseño de interfaces gráficas, implementación de la aplicación, acceso a BD. Además Qt tiene una ventaja sobre otros sistema y es la inclusión de **CSS** para cambiar el estilo visual de los distintos widgets de un formulario. Esto hace que se puedan desarrollar formularios rápidamente con un gran acabo profesional.

6.3.3. Estilo visual de interfaces

Como se puede observar en 6.1 se ha diseña un diseño del estilo visual que deben llevar las interfaces a crear. Esto puede sufrir variaciones según los requisitos necesarios.

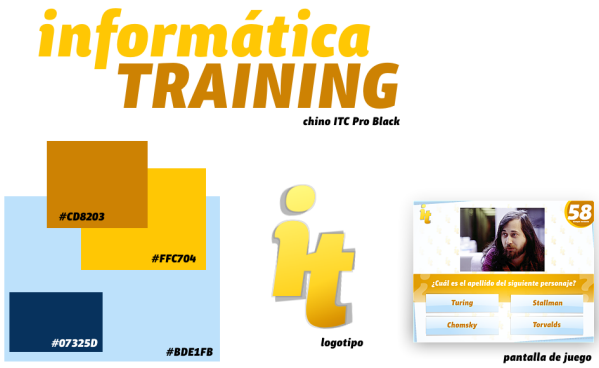

Figura 6.1: Estilo visual interfaces

6.3.4. Documentación del código

Para la documentación de código me apoyaré en Doxygen. Es de gran ayuda por que una vez terminada la aplicación genera un documento muy completo y podemos verificar la correcta implementación de la aplicación.

6.3.5. Sistema de control de versiones

Para llevar un control y además usar un poco a modo de copia de seguridad utilizaré svn. He decidido obtar por esta herramienta y no git (que está extendiéndose muy rápido y que ha demostrado tener mejores prestaciones que svn) por ser un único analista, diseñador y programador. Entonces con svn obtengo todo lo necesario.

6.3.6. Generación de documentación técnica

Para todo la generación de documentación y especificaciones he utilizado latex por su sencillez y por su gran acabado profesional de documentos.

6.3.7. Sistema de gestión de bases de datos

Para el sistema de gestión de bases de datos se ha optado por SQLite3. Este sistema de gestión de bases de datos tiene varias ventajas y son las siguientes:

- Escrito en C++.
- Todos los datos están en un único fichero que se puede exportar entre sistemas.
- Es una aplicación que ocupa poco espacio.
- Realiza operaciones más rápido que mysql.
- No necesita configuración ninguna.
- Integración con qt.
- Multisistema.

Por todas las ventajas anteriores me he decidido por este sistema de gestión de bases de datos al cubrir los requisitos necesarios para la aplicación.

6.4. Interfaz gráfica

Tomando los resultados obtenidos en la fase de análisis es necesario diseñar una interfaz gráfica amigable para el usuario y desde la cual se pueda interactuar con la aplicación. Para el diseño de las interfaces se intentará en todo momento que sean usables además de intentar conseguir que el usuario no pueda introducir datos erróneas para que no produzca comportamientos anómalos.

6.4.1. Pantallas y funcionalidades

A continuación describiré todas las interfaces gráficas que tendrá el sistema así como la funcionalidad que tendrán. Además le asignaré una abreviatura para poder luego identificarlas más fácilmente como para realizar diagramas de interacción mas comprensibles.

- **Portada:** Se administrará los usuarios (crear, borrar) así como seleccionar el usuario con el que vamos a actuar en la aplicación.
- **Estadísticas:** Se podrá observar información relativa a las partidas realizadas por el usuario seleccionado (Datos descriptivos y widget gráfico).
- **Evolución:** Se podrá ver un widget gráfico con la evolución de las últimas 10 partidas realizadas.

- **Playlist:** Administración de la lista de reproducción.

- **Opciones:** Configuración de las opciones de partida así como exportación a PDF.

- **Crear preguntas:** Utilidad para crear preguntas de forma interactiva.

- **Gestión de preguntas:** Utilidad para la administración (Edición, Eliminación) de preguntas dadas de alta en el sistema.

- **Gestión de materias:** Utilidad para la administración (Creación, Edición, Eliminación) de categorías y materias.

- **Importar:** Desde aquí se podrá subir un fichero xml con informacíon para crear preguntas, categorías, materias.

- **Partida:** Se mostrará una interfaz amena para la realización del cuestionario.

- **Resultados:** Pequeño resumen de los resultados obtenidos en el test.

- **Ver resultados:** Se podrá ver las preguntas y respuestas realizada por el usuario.

Por último comentar que se tendrán interfaces gráficas intermedias que agruparán opciones comunes. Esto se puede observar mejor en la siguiente sección donde está el diagrama de interacción.

6.5. Diagrama de interacción entre interfaces gráficas

En el siguiente diagrama podemos observar la interacción entre las distintas interfaces gráficas:

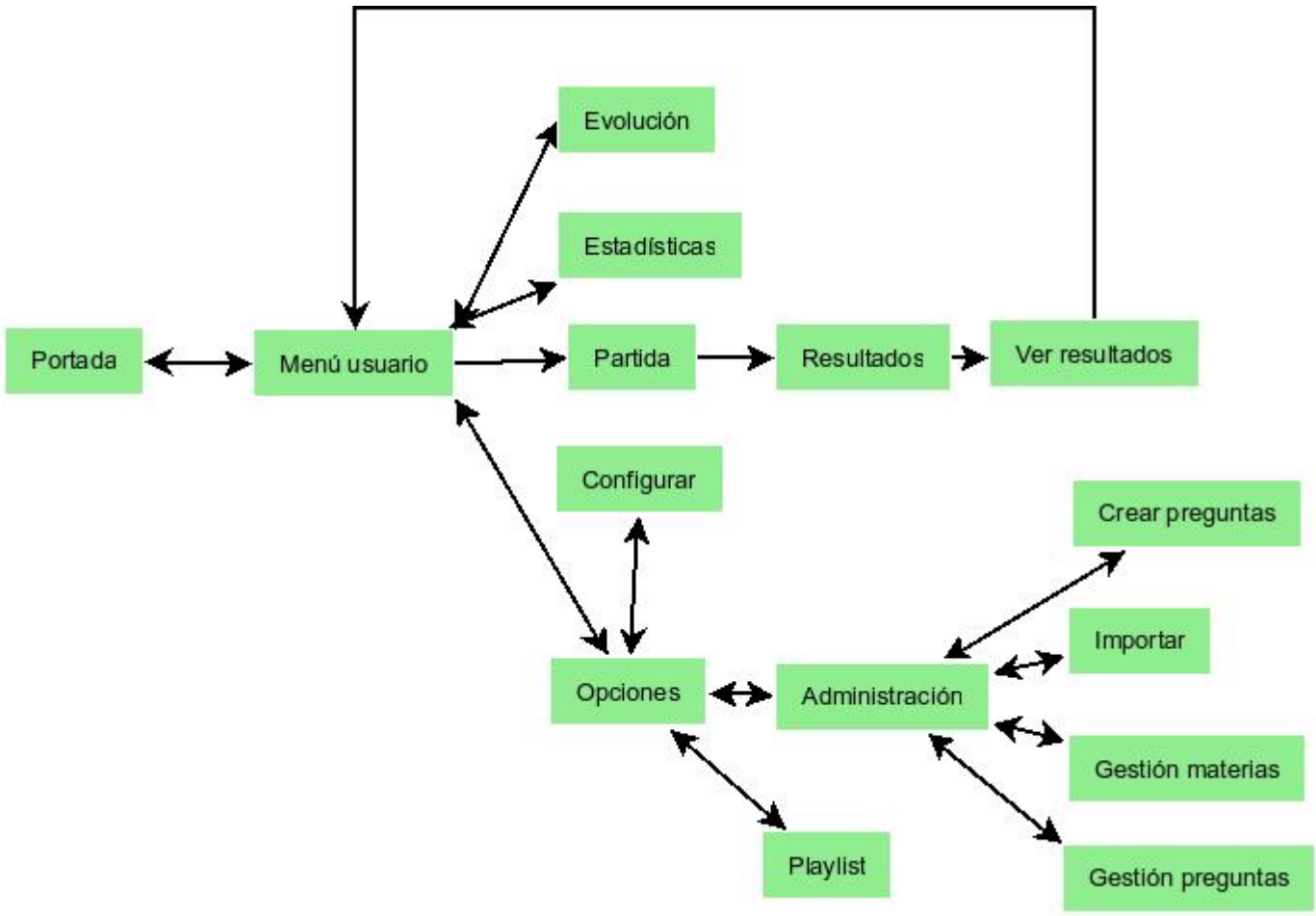

Figura 6.2: Digrama interacción interfaces gráficas

6.6. Diseño diagrama Entidad Relación

En el siguiente diagrama 6.3 se puede observar el diseño de la BD a través de un diagrama ER(Entidad-relación).

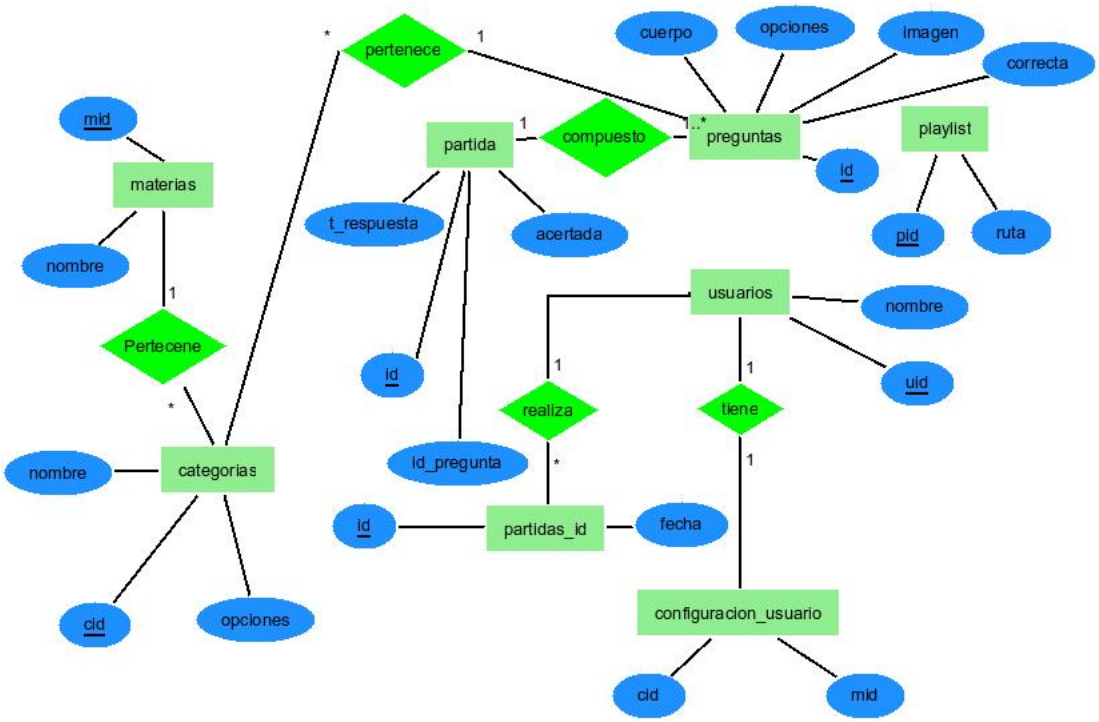

Figura 6.3: Digrama entidad relación

6.7. Descripción de la base de datos

6.7.1. Introducción

En esta sección describiré la base de datos asociada el diagrama ER anterior. Para cada table diré sus campos, claves primarias y valores por defecto.

6.7.2. Tabla categorias

Esta tabla está compuesta por los siguientes campos:

- **cid** tipo: *INTEGER*, clave primaria.
- **mid** tipo: *INTEGER*.
- **nombre** tipo: *TEXT*.
- **opciones_num_preguntas** tipo *INTEGER*, valor por defecto *10*.
- **opciones_valor_erronea** tipo *INTEGER*, valor por defecto *3*.
- **opciones_tiempo** tipo *INTEGER*, valor por defecto *100*.
- **opciones_erroneas_sin_contestar** tipo *INTEGER*, valor por defecto *0*.
- **opciones_partida_sin_tiempo** tipo *INTEGER*, valor por defecto *0*.
- **opciones_verdadero_falso** tipo *INTEGER*, valor por defecto *0*.

6.7.3. Tabla configuracion_usuario

Esta tabla está compuesta por los siguientes campos:

- **uid** tipo: *INTEGER*, clave primaria.
- **mid** tipo: *INTEGER*.
- **cid** tipo: *INTEGER*.

6.7.4. Tabla materias

Esta tabla está compuesta por los siguientes campos:

- **mid** tipo: *INTEGER*, clave primaria.
- **nombre** tipo: *TEXT*.
- **opciones_num_preguntas** tipo *INTEGER*, valor por defecto *10*.
- **opciones_valor_erronea** tipo *INTEGER*, valor por defecto *3*.
- **opciones_tiempo** tipo *INTEGER*, valor por defecto *100*.
- **opciones_erroneas_sin_contestar** tipo *INTEGER*, valor por defecto *0*.
- **opciones_partida_sin_tiempo** tipo *INTEGER*, valor por defecto *0*.
- **opciones_verdadero_falso** tipo *INTEGER*, valor por defecto *0*.

6.7.5. Tabla partida_id

Esta tabla está compuesta por los siguientes campos:

- **id** tipo: *INTEGER*, clave primaria.
- **fecha_partida** tipo: *INTEGER*.
- **uid** tipo: *INTEGER*.

6.7.6. Tabla partidas

Esta tabla está compuesta por los siguientes campos:

- **id** tipo: *INTEGER*, clave primaria.
- **id_partida** tipo: *INTEGER*.
- **id_pregunta** tipo: *INTEGER*.
- **acertada** tipo *INTEGER*.
- **tiempo_respuesta** tipo *INTEGER*.

6.7.7. Tabla playlist

Esta tabla está compuesta por los siguientes campos:

- **pid** tipo: *INTEGER*, clave primaria.
- **ruta** tipo: *VARCHAR*.

6.7.8. Tabla preguntas

- **id** tipo: *INTEGER*, clave primaria.
- **cuerpo** tipo: *VARCHAR*.
- **opcion_a** tipo: *VARCHAR*.
- **opcion_b** tipo *VARCHAR*.
- **opcion_c** tipo *VARCHAR*.
- **opcion_d** tipo *VARCHAR*.
- **imagen** tipo *VARCHAR*.
- **correcta** tipo *INTEGER*.
- **cid** tipo *INTEGER*.

6.7.9. Tabla usuarios

Esta tabla está compuesta por los siguientes campos:

- **uid** tipo: *INTEGER*, clave primaria.
- **nombre** tipo: *TEXT*.
- **score** tipo: *INTEGER*.

6.8. Diagrama de clases asociado al diseño

El siguiente diagrama 6.4 es el obtenido una vez realizado el diseño de las clases. Se ha omitido los campos y funciones para no obtener una diagrama de gran amplitud y poca claridad. Si se quiere extender la documentación podemos acceder a la documentación dentro de la aplicación obtenida con doxygen que es más extensa.

6.8.1. Nuevas clases añadidas

En este apartado comentaré las nuevas clases que se han añadido y que podemos observar en 6.4. Ha de comentarse que por cada cada interfaz gráfica se han generado dos clases. Las que comienzas por ui son las generadas por la aplicación de diseño de interfaces de Qt y las demás son las que implementas las **SLOTS** asociados a las señales que genere el formulario:

- **formgestionpreguntas:** Esta será la clase correspondiente a la interfaz de la administración de preguntas.

- **Ui::FormGestionPreguntas:** formulario generado por la aplicación de diseño de interfaces Qt para la gestión de preguntas.
- **formedicion:** En esta clase se gestiona la edición de preguntas ya creadas en el sistema además de borrarlas. Hay que tener encuenta que una vez borrada una pregunta no se podrá recuperar al menos que se cree exactamente igual. Además gestiona un filtro por materias y categorías para localizar más fácilmente las preguntas que se quieren editar.
- **Ui::FormMenuEdicion:** formulario generado por la aplicación de diseño de interfaces Qt para la gestión de preguntas.
- **formresultado:** En esta clase se gestionará los resultados obtenidos en el test realizado y calculará la puntuación para poder mostrársela adecuadamente al usuario.
- **Ui::FormResultado:** formulario generado por la aplicación de diseño de interfaces Qt para la visualización del resultado de un test.
- **formopciones:** En esta clase se gestiona la configuración global del usuario así como la llamada a la clase de generación de PDF al pulsar el botón PDF.
- **Ui::FormOpciones:** formulario generado por la aplicación de diseño de interfaces Qt para la gestión de las opciones de la aplicación.
- **formenusuario:** En esta clase se gestiona al formulario que se debe enviar una vez pulsado algún botón.
- **Ui::FormMenusuario:** formulario generado por la aplicación de diseño de interfaces Qt para la gestión del menú del usuario.
- **formcrearpreguntas:** En esta clase se gestiona la creación de preguntas comprobando que los datos sean correcto además de controlar que los datos introducidos no puedan sear mayor de su tamaño.
- **Ui::FormCrearPreguntas:** formulario generado por la aplicación de diseño de interfaces Qt para crear preguntas.
- **formconfigurar:** En esta clase se gestiona al formulario que se debe dirigir una vez que el usuario pulse un botón.
- **Ui::FormConfigurar:** formulario generado por la aplicación de diseño de interfaces Qt para la gestión del menú configurar.
- **formain:** en esta clase se gestiona las llamadas a la creacción, eliminación y selección de usuarios.
- **Ui::formMain:** formulario generado por la aplicación de diseño de interfaces Qt para la gestión del menú principal.
- **formplaylist:** En esta clase se gestióna la introducción de canciones para añadirlas a la playlist. Además se comprueba de que los ficheros estén en formato mp3 para un buen uso de la playlist.
- **Ui::Formplaylist:** formulario generado por la aplicación de diseño de interfaces Qt para la gestión de la playlist.
- **formestadisticas:** En esta clase se gestiona todo la relacionado con las estadísticas para poder mostrarlas de una forma entendible al usuario. Es la encargada de llamar a la clase estadísticas que realiza las extracciones de los datos de la BD para devolvérsela y mostrarlos.

- **Ui::FormEstadisticas:** formulario generado por la aplicación de diseño de interfaces Qt para la visualización de estadísticas.
- **formgestionmagerias:** En esta clase se gestiona la creación y eliminación de materias y categorías. Ha que tener encuenta que al borrar una materia o categoría se borrarán todas sus preguntas asociadas.
- **Ui::FormGestionMaterias:** formulario generado por la aplicación de diseño de interfaces Qt para la gestión de materias y categorías.
- **formverespuestas:** En esta clase se gestiona la visualización y navegación por los resultados obtenidos en el test realizado.
- **Ui::FormVerRespuestas:** formulario generado por la aplicación de diseño de interfaces Qt para la visualización de las respuestas realizadas en el test.
- **formimportar:** En esta clase se gestiona la subida del fichero con los datos a importar. Además se encarga de llamar a las clases necesarias para la creación de preguntas, materias y categorías.
- **Ui::Formimportar:** formulario generado por la aplicación de diseño de interfaces Qt para la gestión de la importación de datos.
- **formtest:** En esta clase se gestiona la visualización y navegación de la realización del test con la configuración seleccionada por el usuario.
- **Ui::FormTest:** formulario generado por la aplicación de diseño de interfaces Qt para la realización de un test.
- **Ui::FormParent:** formulario generado por la aplicación de diseño de interfaces Qt para englobar a todas las clases.
- **formnopreguntas:** En esta clase se gestiona el mensaje que se debe mostrar cuando se intenta realizar un test que no contiene pregunta.
- **Ui::FormNoPreguntas:** formulario generado por la aplicación de diseño de interfaces Qt que muestra un mensaje cuando no hay preguntas para realizar el test.
- **formpadre:** En esta clase se gestiona el cambio de formularios así como engloblar a todos los formularios. En esta clase está implementando un pequeño autómata que es el encarga del buen funcionamiento del la interfaz gráfica 6.2. Para ello se asociará un número a cada formulario para poder identificarlo y saber más fácilmente como cambiar entre uno y otro. Esta asociación es la siguiente:
  - formain ->form1
  - formenusuario ->form2
  - formtest ->form3
  - formresultado ->form4
  - formverespuestas ->form5
  - formestadisticas ->form6
  - formconfigurar ->form7
  - formplaylist ->form8
  - formopciones ->form9

- formedicion ->form10
- formgestionmaterias ->form11
- formgestionpreguntas ->form12
- formnopreguntas ->form13
- formcrearpreguntas ->form14
- formimportar ->form15

6.9. Clases del núcleo de la aplicación

Al ser programación orientada a eventos y tener muchos eventos creo que es innecesario la realización de un diagrama de secuncia del sistema por que quedaría poco explicativo además de poder provocar confusiones luego en la fase de implementación en consecuencia de esto comentaré las clases que conforman el núcleo de la aplicación además de explicar interactúan con los formularios:

- **game:** En esta clase se englobal toda la aplicación. Es la encarga de gestionar los idiomas, usuarios y playlist. Además es la que llama a *formpadre* para que empieze la ejecución de los formularios. A su vez formpadre al llamarse por primera vez llamará a *formain* que es desde donde se gestionan los usuarios.

- **estadisticas:** En esta clase se realizan las consultas necesarias a la base de datos para obtener datos estadísticas sobre las partidas de los usuarios. Con esta clase interactúa el formulario *formestadisticas*.

- **gestionmaterias:** Esta clase es la encargada de gestionar (creación, eliminación) de materias y categorías así como proporcionar utilidades relacionadas con las materias y categorías. Los formularios que interactúan con ella son *formedicion, formopciones, formcrearpreguntas, formimportar, formgestionpreguntas, formgestionmaterias.*

- **gestionpreguntas:** Esta clase es la encargada de gestionar las preguntas (creación, eliminación y modificación) así como proporcionar utilidades relacionadas con ls preguntas. Será también la encargada de generar el PDF. Los formularios que interactúan con ella son *formedicion, formopciones, formcrearpreguntas, formimportar, formgestionpreguntas, formgestionmaterias.*

- **partida:** Esta clase es la encargada de generar los test así como de controlar las respuestas y generar la puntuación obtenida. Además se encarga de introducir los datos de la partida en la base de datos para poder ser luego utilizados en las estadísticas. Los formularios que realizan uso de ella son *formtest, formresultado, formverespuestas.*

- **pieview:** Esta clase es la encarga de realizar un gráfico de tarta a través de los datos que le son pasados. las formularios que hacen uso de ella son *formestadisticas.*

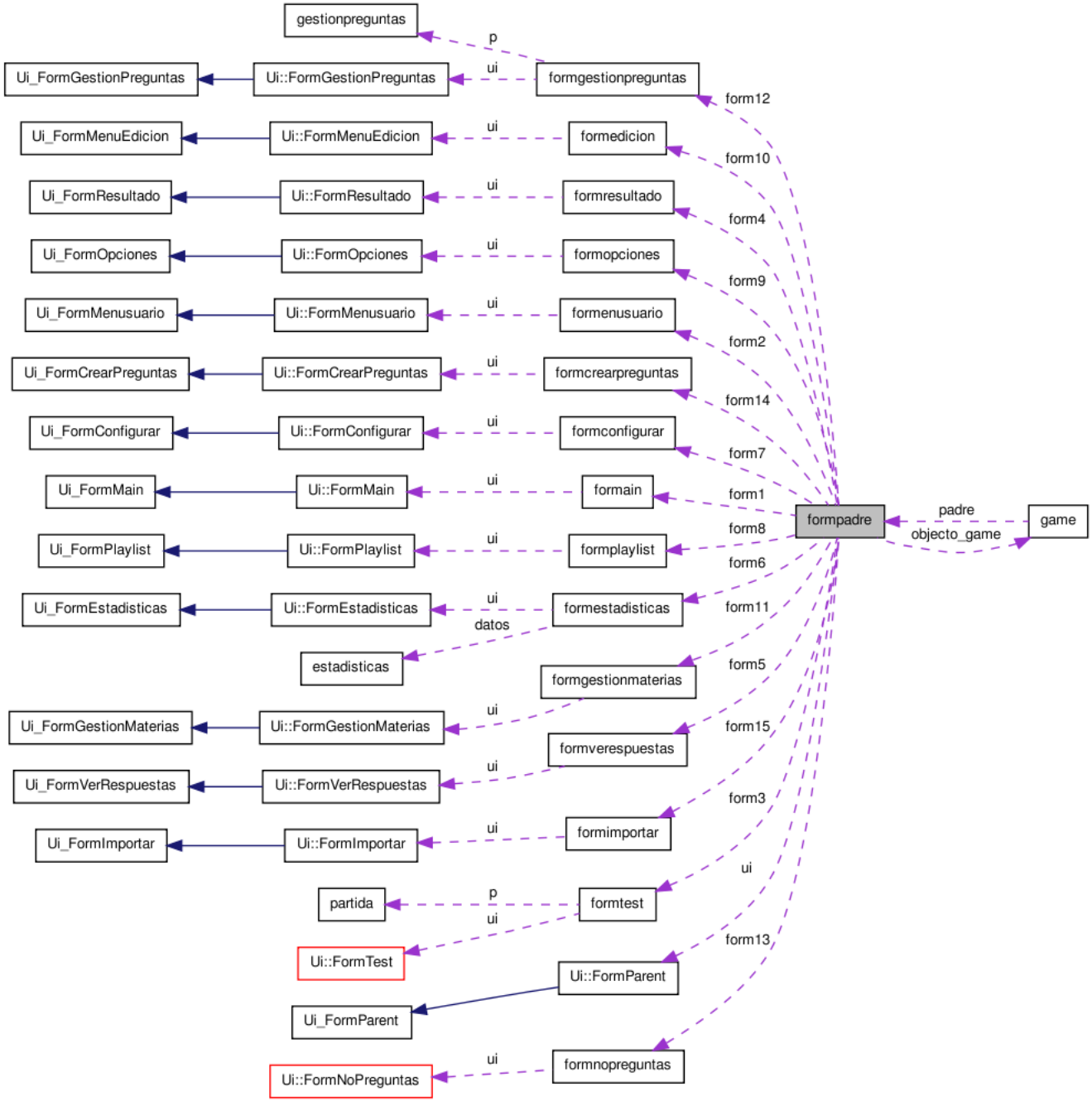

Figura 6.4: Diagrama de clases asociado al diseño

Capítulo 7

Implementación

7.1. Introducción

Al ser una aplicación muy extensa en código en esta sección comentaré código de especial relevancia así como asuntos de interés especial. Además he de hacer mención a que todo el código implementado está documentado en base a Doxygen teniendo todo la documentación generada disponible en la aplicación.

7.2. Guía de estilo

7.2.1. Introducción

En esta sección describiré las reglas escritura de código para que sea más legible y para que futuros desarrolladores puedan usarla de una forma cómoda y sencilla.

7.2.2. Estructuras de control

- Estructura de control **if**:

```
        if (condition1 || condition2) {
                action1;
        }
        else if (condition3 && condition4) {
                action2;
        }
```

- Estructura de control **switch**:

```
        switch (condition) {
                case 1:
                action1;
        break;

                case 2:
                action2;
        break;

                default:
        defaultaction;
        }
```

7.2.3. Llamadas a funciones

```
1          int var = funcion(bar, baz, quux);
```

7.2.4. Declaración de funciones

```
1    void funcion(int a) {
2      int valor = a;
3      return valor;
4    }
```

7.2.5. Concatenación de cadenas

```
1    QString string = "Foo"+ variable;
2    QString string = variable +"foo";
3    QString string = funcion() + "foo";
4    QString string = "foo"+"bar";
```

7.3. Comienzo de la implementación

Para comenzar con la aplicación primero debemos crear un fichero de configuración para Qt.Este fichero con extensión *pro* tendrá el siguiente contenido:

```
TEMPLATE = app
TRANSLATIONS = translates/ingles.ts
TRANSLATIONS += translates/italiano.ts
TRANSLATIONS += translates/aleman.ts
TRANSLATIONS += translates/espanol.ts
unix:LIBS += -lsqlite3
QT += core sql phonon
CONFIG += uitools
OBJECTS_DIR = objects
MOC_DIR = objects
UI_DIR = objects
unix {
 TARGET = informatica_training
 OBJECTS_DIR +=  build/o/unix
 database.path += ~/bin
 database.files = BD/BD.sqlite
 target.path += ~/bin
 INSTALLS += database \
  target
}
```

```
FORMS += forms/formain.ui forms/formenusuario.ui forms/formparent.ui
 forms/formtest.ui forms/form_resultado.ui forms/form_ver_respuestas.ui
 forms/formestadisticas.ui forms/formenuedicion.ui
  forms/formgestionmaterias.ui forms/formgestionpreguntas.ui
   forms/formconfigurar.ui forms/formplaylist.ui
  forms/formopciones.ui forms/form_no_preguntas.ui
   forms/formcrearpreguntas.ui forms/formimportar.ui
HEADERS += includes/formain.h includes/game.h
 includes/formenusuario.h includes/formpadre.h
  includes/formtest.h includes/partida.h
   includes/formresultado.h
 includes/formverespuestas.h includes/formestadisticas.h
  includes/pieview.h includes/estadisticas.h includes/formedicion.h
   includes/formgestionmaterias.h
  includes/formgestionpreguntas.h includes/gestionmaterias.h
   includes/gestionpreguntas.h includes/formconfigurar.h
    includes/formplaylist.h
   includes/formopciones.h includes/formnopreguntas.h
    includes/formcrearpreguntas.h includes/formimportar.h
     includes/xmlstreamreader.h includes/xmlParser.h includes/utils.h
SOURCES += src/main.cpp src/game.cpp src/formain.cpp
 src/formenusuario.cpp src/formpadre.cpp src/formtest.cpp
 src/partida.cpp src/formresultado.cpp src/formverespuestas.cpp
  src/formestadisticas.cpp src/pieview.cpp src/estadisticas.cpp
  src/formedicion.cpp src/formgestionmaterias.cpp
   src/formgestionpreguntas.cpp src/gestionmaterias.cpp
    src/gestionpreguntas.cpp
   src/formconfigurar.cpp src/formplaylist.cpp
   src/formopciones.cpp src/formnopreguntas.cpp src/formcrearpreguntas.cpp
    src/formimportar.cpp
    src/xmlstreamreader.cpp src/xmlParser.cpp src/utils.cpp
```

Explicaré las partes más importantes de este fichero y sus funciones:

- **TRANSLATIONS**: En esta variable se añadirán los ficheros donde se almacenan las traducciones. En este caso inglés, italiano y español.
- **QT**: En esta variable almacenaremos las librerías de las que vamos a hacer uso. En este caso vamos a hacer uso del core de Qt, integración con sql y phonon que es una librería multimedia para el uso del sonido.
- **OBJECTS_DIR, MOC_DIR, UI_DIR** Estas variables se usan para decirle a aplicación donde almacenar los objetos de la compilación así con los ficheros obtenidos de los formularios.
- **unix**: Aquí colocamos la configuración para los sistemas unix. En el caso anterior el ejecutable se llamará informatica_training, y la base de datos BD.sqlite
- **FORMS**: En esta variable se coloca los formularios que vallamos a usar.
- **HEADERS**: En esta variable se coloca la declaración de las clases.
- **SOURCES**: En esta variable colocamos los fuentes de la aplicación.

Una vez realizado el fichero de configuración bastará ejecutar *qmake* en el directorio de la aplicación que que nos genere el makefile necesario para poder ejecutarla.

7.4. Ejemplos de implementación

7.4.1. XML asociado a la importación

El siguiente código muestra un xml sencillo que acepta la importación de la aplicación:

```
<?xml version="1.0" encoding="UTF-8"?>
<Importacion>
  <Materias>
    <Materia>Prueba1</Materia>
    <Materia>Prueba2</Materia>
  </Materias>
  <Categorias>
    <Categoria>
      <Nombre>Prueba1-1</Nombre>
      <Materia>Prueba1</Materia>
    </Categoria>
<Categoria>
      <Nombre>Prueba2-1</Nombre>
      <Materia>Prueba2</Materia>
    </Categoria>
  </Categorias>
  <Preguntas>
    <Pregunta>
      <Cuerpo>¿Lorem ipsum?</Cuerpo>
<ID_Materia>Prueba1</ID_Materia>
      <ID_Categoria>Prueba1-1</ID_Categoria>
      <a>Lorem</a>
      <b>Lorem</b>
      <c>Lorem</c>
      <d>Lorem</d>
<Correcta>a</Correcta>
      <Ruta_imagen>/home/fastangel/Desktop/Pantallazo-1.png</Ruta_imagen>
    </Pregunta>
  </Preguntas>
</Importacion>
```

Como se observa en el **XML** el documento a importar se divide en tres secciones:

- En la primera agrupado por la etiqueta *Materias* colocamos las materias nuevas que vamos a crear. En caso de que existan la aplicación no realizará ninguna acción.
- En la segunda agrupado por la etiqueta *Categorias* colocamos las categorías nuevas que vamos a crear. Como sucede con las materias si la categoría ya existe no se realizará ninguna acción.
- Por último agrupado por la etiqueta *Preguntas* colocamos las preguntas nuevas que vamos a crear.

7.4.2. Redimensionado y escalado de imágenes

Al llevar las preguntas imágenes asociadas surgió el problema de si tener las rutas de las imágenes para asociarlas o en cambio copiarlas para no tener problemas cuando el usuario la cambie de lugar o la borre. Como se ha obtado por copiar y escalar las imágenes el código que lo realiza es el siguiente:

```
    QString nueva_ruta;
    uint i = 0, ancho_sobrante = 441, altura_sobrante = 121, width, height;
    bool status;
    QFileInfo fi(ruta_img);
    QFileInfo nuevo_file;
    QPixmap original(ruta_img);
    original.scaled(441, 121, Qt::KeepAspectRatioByExpanding);
    width = original.width();
    height = original.height();
    if (width > 441) {
      ancho_sobrante += (uint)(width - 441) / 2;
    }
    if (height > 121) {
      altura_sobrante += (uint)(height - 121) / 2;
    }
    QPixmap final = original.copy(ancho_sobrante - 441, altura_sobrante -
       121, 441, 121);
    do {
      nueva_ruta = QApplication::applicationDirPath() +"/img_preguntas/"+
         QString::number(i) +"_"+ fi.fileName();
      nueva_ruta_sql = "/img_preguntas/"+ QString::number(i) +"_"+ fi.
         fileName();
      nuevo_file = QFileInfo(nueva_ruta);
      status = nuevo_file.exists();
      if (!status) {
        final.save(nueva_ruta);
      }
      i++;
    } while(status);
```

Como se puede ver en el código primero escalamos la imagen a 441 x 121 pero mantenimiento el aspect ratio.

```
original.scaled(441, 121, Qt::KeepAspectRatioByExpanding);
```

Una vez escalada una imagen pasamos a obtener los píxeles sobrantes de la altura y anchura para realizar un crop y dejar la imagen con el tamaño adecuado (Este recorte se realiza partiendo desde el centro para que la imagen quede centrada).

```
  width = original.width();
  height = original.height();
  if (width > 441) {
    ancho_sobrante += (uint)(width - 441) / 2;
  }
  if (height > 121) {
    altura_sobrante += (uint)(height - 121) / 2;
  }
  QPixmap final = original.copy(ancho_sobrante - 441, altura_sobrante -
     121, 441, 121);
```

Por último lo único que queda es guardar la imagen. Para esto se ha optado por un método para que no ocurra problemas con imágenes con el mismo nombre. Para ello si existe el nombre la imagen se copiara con el siguiente nombre *nombre_x*. Esto se realiza con:

```
do {
  nueva_ruta = QApplication::applicationDirPath() +"/img_preguntas/"+
     QString::number(i) +"_"+ fi.fileName();
  nueva_ruta_sql = "/img_preguntas/"+ QString::number(i) +"_"+ fi.
     fileName();
  nuevo_file = QFileInfo(nueva_ruta);
  status = nuevo_file.exists();
  if (!status) {
    final.save(nueva_ruta);
  }
  i++;
} while(status);
```

7.4.3. Lectura de XML

Para la lectura de XML se ha optado por una librería sencilla y de fácil manejo. Un ejemplo de lectura es:

```
XMLNode XMLscene;
XMLscene = XMLNode::openFileHelper(fileName.toStdString().c_str(), "
   importacion");
XMLNode xMaterias = XMLscene.getChildNode("Materias");

int n = xMaterias.nChildNode("Materia");
      QString nombre_materia;
for (int i=0; i < n; i++) {
  nombre_materia = xMaterias.getChildNode("Materia", i).getText();
              gestionmaterias m(nombre_materia);
}
```

Como se observa con:

```
XMLscene = XMLNode::openFileHelper(fileName.toStdString().c_str(), "
   importacion");
```

Abrimos el xml y obtenemos la raiz que es importación. Una vez leido la raiz pasamos a leer todos los hijos que en este caso son Materia. Además a la vez que lo vamos leyendo se irán creando:

```
XMLNode xMaterias = XMLscene.getChildNode("Materias");

int n = xMaterias.nChildNode("Materia");
      QString nombre_materia;
for (int i=0; i < n; i++) {
  nombre_materia = xMaterias.getChildNode("Materia", i).getText();
              gestionmaterias m(nombre_materia);
}
```

7.4.4. Gestión de audio

Para el tratamiento del audio se ha optado por realizarlo a través de la librería **Phonon** que viene integrada con Qt y que da un rendimiento bueno.

7.4.5. Traducciones

Para la internacionalización he aprovechado la utilidad que proporciona Qt. La forma de usarlo es la siguiente:

```
1    QTranslator *translator = new QTranslator();
2    translator->load(QDir::toNativeSeparators(dir_app+"/translates/espanol.qm
        "));
```

A través de la clase *QTranslator* instalamos los idiomas que vamos a usar. Esto se realiza especificando el fichero de traducción y Qt se encargará de leer la configuración del fichero para averiguar el idioma. Cuando se quiere cambiar de idioma lo único que se tiene que realizar es volver a llamar al método *load* de la clase QTranslator.

7.4.6. Conexión con Base de datos

La conexión con la BD se hace globalmente y únicamente debe realizarse una vez. Una vez conectada podremos consultarla durante todo el periodo de ejecución. El siguiente código es el que se utiliza para la conexión:

```
1    QString dir_app =  QApplication::applicationDirPath();
2    QSqlDatabase basedatos = QSqlDatabase::addDatabase("QSQLITE");
3    basedatos.setDatabaseName(QDir::toNativeSeparators(dir_app+"/BD/BD.sqlite
        "));
4    if (!basedatos.open()) {
5      QMessageBox::critical(this,"Base de Datos no abierta", "La Base de
          Datos no puede ser abierta", QMessageBox::Ok);
6    }
```

7.4.7. Implementación de formularios

Por cada formulario diseñado con Qt designer se nos generará una cabecerá con toda la información. Luego se tiene que implementar una clase que haga uso de estos datos además de implementar los slots asociados a las señales de los objetos de los formularios. Comentar que los SLOTS son las funciones que se ejecutan cuando una determinada señal es enviada a un elemento del formulario. Un ejemplo de uso de esta cabecerá es la siguiente clase:
formenusuario.h

```
1  #ifndef FORMENUSUARIO_H
2  #define FORMENUSUARIO_H
3
4  #include "ui_formenusuario.h"
5
6  class formenusuario : public QWidget {
7          Q_OBJECT
8
9          public:
```

```
        formenusuario(QWidget *parent = 0);
    void inicializar(QString);
        private slots:
    void comenzar_test();
    void configurar_aplicacion();
    void atras_main();
    void estadisticas_usuario();
        private:
    Ui::FormMenusuario ui;
};

#endif
```

formenusuario.cpp

```
#include <QtUiTools>
#include <QtGui>
#include <QString>

#include "../includes/formenusuario.h"
#include "../includes/formpadre.h"


/**
 * Constructor que inicializa la interfaz del formulario form menu usuario
 */
formenusuario::formenusuario(QWidget *parent): QWidget(parent) {
  ui.setupUi(this);

  formpadre *aux = (formpadre*) this->parentWidget();
  game *objecto_partida = aux->obtener_partida();
  objecto_partida->game_inicializar_configuracion_usuario();
}

/**
 * Metodo para inializar el label nombre de usuario
 */
void formenusuario::inicializar(QString name) {
  ui.label_nombreusuario->setText(name);
}

/**
 * SLOT que se activa al pulsar comenzar test.
 */
void formenusuario::comenzar_test() {
  formpadre *aux = (formpadre*) this->parentWidget();
  aux->formpadre_change_scene(3,2);
}

/**
 * SLOT que se activa al pulsar configurar aplicacion.
 */
void formenusuario::configurar_aplicacion() {
formpadre *aux = (formpadre*) this->parentWidget();
  aux->formpadre_change_scene(7,2);
```

```
41  }
42
43  /**
44   * SLOT que se activa al pulsar atras.
45   */
46  void formenusuario::atras_main() {
47    formpadre *aux = (formpadre*) this->parentWidget();
48    aux->formpadre_change_scene(1,2);
49  }
50
51  /**
52   * SLOT que se activa al pulsar estadisticas.
53   */
54  void formenusuario::estadisticas_usuario() {
55    formpadre *aux = (formpadre*) this->parentWidget();
56    aux->formpadre_change_scene(6,2);
57  }
```

Capítulo 8

Pruebas

En este apartado describiré las pruebas realizas a la aplicación y los resultados obtenidos. Al realizar una misma persona la parte de análisis, diseño e implementación he solicitado a terceras personas que prueben la aplicación para que los datos no sean sesgados y que la aplicación sea más fiable.

8.1. Introducción

Lo primero que he realizado antes de realizar un plan de pruebas es verificar uno a uno los requisitos iniciales del proyecto para ver si se han cumplido o no. Satifastoriamente se cumplen todos los objetivos iniciales. Una vez verificado los requisitos iniciales pasamos a diseñar un plan de pruebas para aplicárselo a la aplicación. Una vez pasado este plan se pasará de nuevo a implementación para corregir los bugs encontrados o fallos del sistema. Si los errores fueran de usabilidad o diseño se volvería a la fase de diseño para posteriormente pasar de nuevo a implementación o corregirlo. Hasta que la aplicación no pase todo el test de prueba no se podrá considerar finalizada. Como no se podrá probar todas las posibles entradas de datos las pruebas se diseñarán intentanto llevar a los valores máximos que se pueden introducir para ver el comportamiento.

8.2. Primera prueba

¿Se comporta adecuadamente la aplicación con cadenas largas dentro de los inputext?
Se encontró un problema con los inputext del nombre de usuario, materias, categorías, cuerpo de preguntas y respuestas a b c d. Al ser muy amplios luego al mostrar los datos en la aplicación no se veían adecuadamente. La solución ha sido limitar la entrada y si superamos el tamaño no se pueda crear usuario, materia, ...

8.3. Segunda prueba

¿Se pueden crear datos duplicados?
Se ha realizado un test comprobando si se pueden crear usuarios con el mismo nombre, materias y categorías. Como en el diseño se tuvo encuenta este tipo de problemas no se dió posibilidad a que ocurrieran estos problemas por lo que esta prueba se ha pasado si ningún problema.

8.4. Tercera prueba

¿Se pueden realizar partidas de materias sin categorías?
Al pasar este test se comprobo que la aplicación fallaba con estos casos. La solución que se tomó en

consecuencia fue el poner un mensaje al intentar realizar el test que avise de que no hay ninguna pregunta y que por favor las cree.

8.5. Cuarta prueba

¿Se pueden realizar partidas sin preguntas?
En este caso ocurre como en la anterior prueba. Si en una partida no hay preguntas se avisará al usuario con un mensaje de que no hay preguntas y que por favor las cree para poder realizar una partida.

8.6. Quinta prueba

¿Qué ocurre si se ha borrado una canción que estaba en la playlist?
Al borrar una canción de la playlist la aplicación no fallaba pero se quedaba almacenada en memoria. La solución para corregir este problema fue el añadir las rutas de las canciones a la base de datos y al iniciar la aplicación comprobar si siguen existiendo o no. En caso de no existir se elimina de la lista de reproducción.

8.7. Sexta prueba

¿La aplicación es amigable y con colores alegres que inviten al juego?
Para esta prueba me he ayudado de personas externas que me han dado se opinión probando la aplicación durante unas horas. Los resultados fueron que les gusto el colorido y tras varias horas de uso no se aburrían y seguían animados a utilizarla.

8.8. Séptima prueba

¿La aplicación es usable?
Para esta prueba me he ayudado de expertos en usabilidad que me han dado se opinión sobre la aplicación. Los resultados han sido cambiar de lugar y tamaño algunos botones para que la interfaz fuera más fácil de manejar y entendible por el usuario.

Capítulo 9

Conclusiones

En este apartado comentaré las experiencias vividas a través de la implementación de la aplicación así como ideas que han surgido durante el desarrollo y que se podrían aplicar en un futuro a la aplicación.

9.1. Opinión personal

El principal problema que me he encontrado en el de desarrollo de la aplicación es en la estimación del tiempo. Es muy difícil estimar el tiempo de un proyecto cuando no se ha realizado nada de esta envergadura. Además también de que se tiene que aprender nuevas herramientas y problemas que surgen en el desarrollo.
Tanto el análisis y el desarrollo han estado dentro de los tiempos que tenía estimados por que es algo muy metódico y fácil de calcular. Una vez que pasé a la implementación el aprendizaje de Qt tiene una curva de aprendizaje muy baja. Es una herramienta muy potente que cada vez que se va aprendiendo más se pueden hacer cosas muy complejas con muy pocas líneas de código. Lo que más me ha gustado es que tiene una gran capa de abstracción (Por ejemplo implementa una versión completa de la STL).
Respecto al diseño de interfaces he aprendido que lleva mucho tiempo hacer una interfaz agradable a la vista y usable es algo que muchas veces no se le dedica el tiempo necesario y que yo diría que es de las cosas más importantes.
Respecto al tema de pruebas me he dado cuenta que cuando una misma persona realizada el análisis, diseño e implementación es muy difícil encontrar bugs por que es muy difícil evaluar el funcionamiento de algo implementado por uno mismo.
Por último comentar que estoy satisfecho con la aplicación obtenida puesto que aunque no sea muy compleja para mí es algo de lo que sentirme orgulloso.

9.2. Posibles mejoras futuras

A continuación comentaré posibles mejoras que han ido surgiendo a lo largo del proyecto o que no estaban especificadas en las especificaciones iniciales:

- Una de las mejoras que considero que la daría más impulso a la aplicación es la de poder exportar las preguntas, materias, categorías, usuarios, estadísticas en xml para poder llevarla a otra aplicación (Aunque esto se puede hacer ahora mismo cambiando la BD pero es más engorroso). Yo pensé en un principio que un usuario pudiera coger la BD y moverla (de ahí la elección de sqlite en vez de otro gestor de bases de datos) pero creo que para usuarios pocos experimentados les puede resultar difícil.
- Algo que sería bastante más ambiciosa sería poder ejecutar la aplicación en una red local a través del cual una persona actuaría de profesor colocando las preguntas y otros como examinados. Así

sería como un examen local a través de ordenadores. Una vez finalizado le saldría el resultado al profesor de todos los examinados. Esta mejorá necesitaría de un estudio de viabilidad puesto que es una gran ampliación y que implicaría no tanto análisis y diseño como implementación.

- Posibilidad de elegir el número de respuestas y poner que pueda haber más de una correcta o ninguna.
- Algo que haría que la aplicación sea más multimedia sería el poder incluir vídeos en las preguntas. Esta ampliación no sería muy dificultosa gracias a la librería utilizada para la reproducción de música **Phonon** que proporciona una capa de abstracción para el manejo de ficheros multimedia (vídeos, música, ...).
- Por último la última ampliación que se me ocurre es poder seleccionar layouts diferences para la aplicación así un usuario podría diseñar la aplicación para sentirse más cómodo mientras la usa.

Capítulo 10

Manual de usuario

10.1. Introducción

En el siguiente manual encontrarás una guía rápida para aprender a usar la herramienta **Informática Training** con la que podrás realizar test de una forma agradable y divertida. Además obtendrás datos de tu evolución así como poder escuchar música de tu ordenador mientras los realizas.
A contuniuación encontrarás las diferentes pantallas por orden que te encontrarás al ejecutar la aplicación.

10.2. Pantalla principal

En la siguiente imagen 10.1 podemos observar la pantalla principal que nos encontraremos al ejecutar la aplicación. En ella podemos seleccionar el usuario con el cual comenzaremos a realizar test o por el contrario borrar y crear un nuevo usuario.

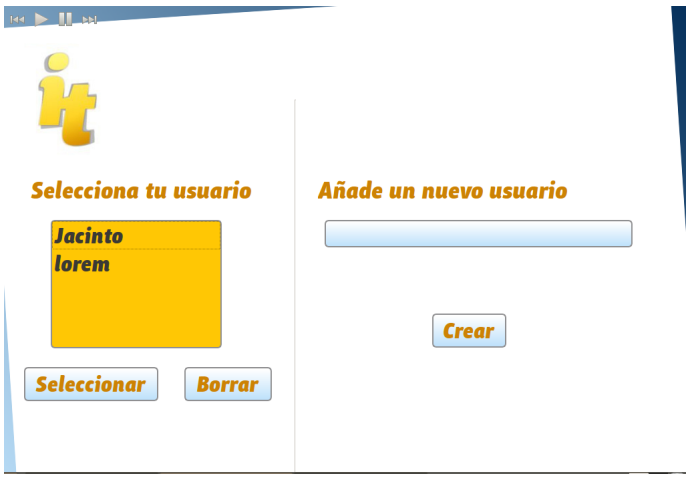

Figura 10.1: Pantalla de entrada

10.3. Pantalla menú de usuario

En la siguiente imagen 10.2 encontramos el menú de usuario desde este menú podremos acceder a las estadísticas, comenzar partida, configurar la aplicación o por el contrario volver a la pantalla principal

para cambiar de usuario.

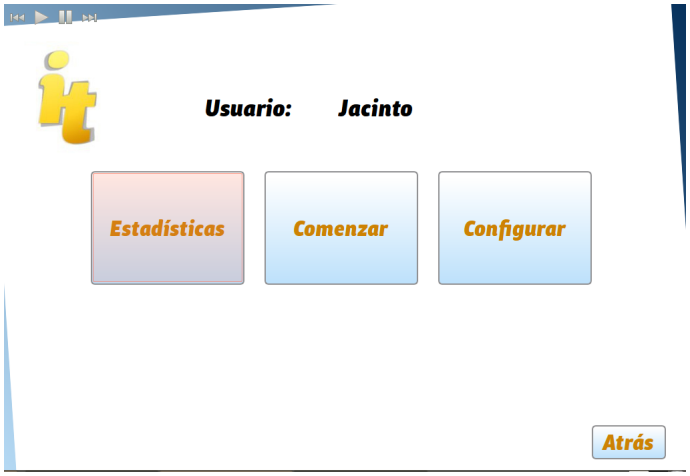

Figura 10.2: Menú de usuario

10.4. Pantalla menú configurar

En la siguiente imagen 10.3 encontramos el menú de configuración. Desde aquí se podrá configurar las opciones de la aplicación, administrar preguntas materias ..., gestionar la playlist y por el contrario volver al menú de usuario.

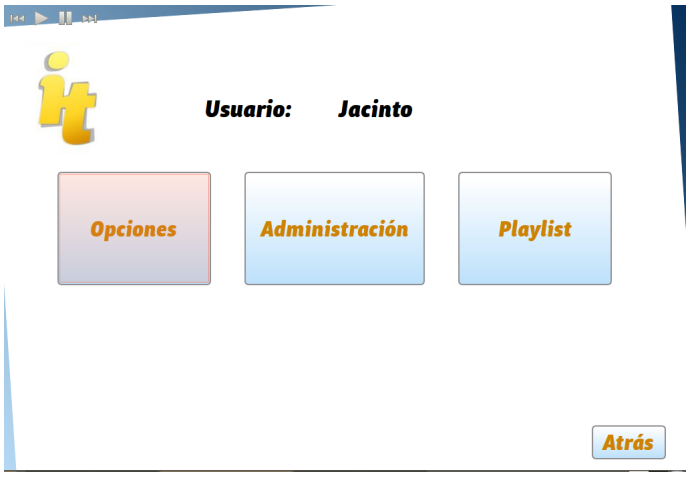

Figura 10.3: Menú configurar

10.5. Pantalla configurar

En la siguiente imagen 10.4 encontramos las opciones. Desde aquí se configura el tipo de partida que se desea realizar quedando guardado para usos posteriores de la aplicación. Además se configurar el idioma y por último también se realiza la exportación a PDF. Para realizarla basta con seleccionar las opciones de partida y pulsar el botón PDF. Una vez pulsado se nos pedirá donde guardar el fichero generado.

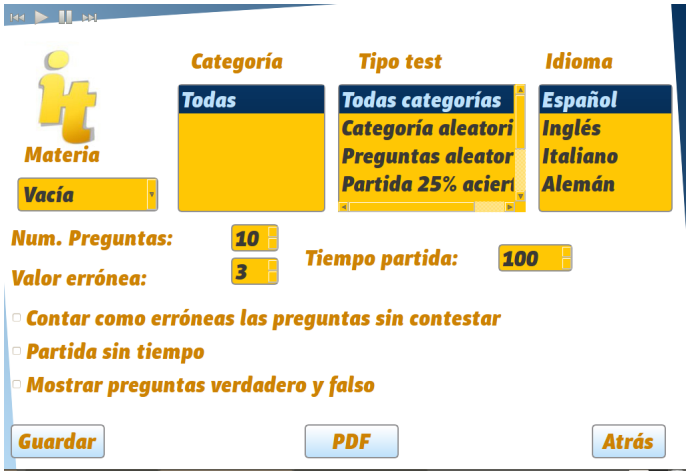

Figura 10.4: Pantalla configurar

10.6. Menú administración

En la siguiente imagen 10.5 encontramos el menú de administración. Desde aquí se selecciona las distintas categorías de administración de la web (gestión y creación de preguntas, gestión de materias y categorías y importación de datos). Además de volver al menú de usuario.

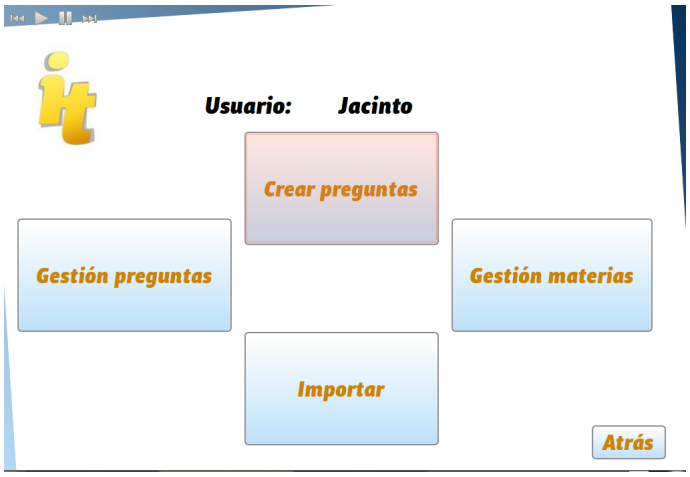

Figura 10.5: Menú administrar

10.7. Creación de pregunta

En la siguiente imagen 10.6 encontramos la creación de preguntas. Desde aquí se puede crear una nueva pregunta. Hay que prestar especial interés a las notas de la izquierda puesto que si no se cumplen estos requisitos no se nos dejará crear la pregunta. En caso de que no se cumplan estos requisitos se nos avisará con un mensaje en color rojo.

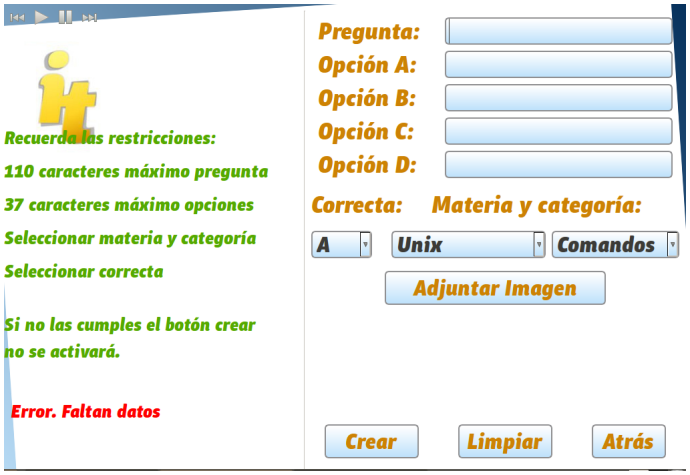

Figura 10.6: Creación de pregunta

10.8. Gestión de preguntas

En la siguiente imagen **??** encontramos la gestión de preguntas. Desde esta interfaz podremos editar las preguntas. He de tenerse en cuenta que hay que cumplir los requisitos que se tenían para la creación de preguntas. Además podremos filtrar por matería y categoría para así ser más fácil la localización de preguntas.

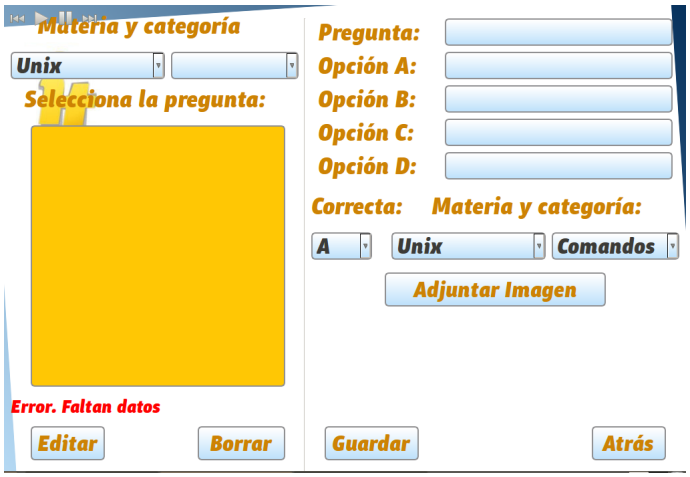

Figura 10.7: Gestión de preguntas

10.9. Gestión de materias

En la siguiente imagen 10.8 encontramos la gestión de materias y categorías. Desde esta interfazar se gestionan las materias y categorías. Hay que tener encuenta que al borrar las materias y las categorías se borrarán las preguntas asociadas no pudiéndose recuperar nunca más.

Figura 10.8: Gestión de materias

10.10. Importación de datos

En la siguiente imagen 10.9 encontramos la importación de datos. Desde esta interfaz importaremos un fichero xml con información relativa para la creación de materias, categorías y preguntas. El formato de este XML debe ser el siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<Importacion>
  <Materias>
    <Materia>Prueba1</Materia>
    <Materia>Prueba2</Materia>
  </Materias>
  <Categorias>
    <Categoria>
      <Nombre>Prueba1-1</Nombre>
      <Materia>Prueba1</Materia>
    </Categoria>
<Categoria>
      <Nombre>Prueba2-1</Nombre>
      <Materia>Prueba2</Materia>
    </Categoria>
  </Categorias>
  <Preguntas>
    <Pregunta>
      <Cuerpo>¿Lorem ipsum?</Cuerpo>
<ID_Materia>Prueba1</ID_Materia>
      <ID_Categoria>Prueba1-1</ID_Categoria>
      <a>Lorem</a>
      <b>Lorem</b>
      <c>Lorem</c>
      <d>Lorem</d>
<Correcta>a</Correcta>
```

```
      <Ruta_imagen>/home/fastangel/Desktop/Pantallazo-1.png</Ruta_imagen>
    </Pregunta>
  </Preguntas>
</Importacion>
```

Figura 10.9: Importación de datos

10.11. Gestión de playlist

En la siguiente imagen 10.10 encontramos la gestión de playlist. Desde esta interfaz añadiremos canciones a nuestra playlist. No hay que preocuparse si se quita alguna canción puesto que el sistema se encargará de eliminarla automáticamente. La única restricción es que solo se pueden añadir ficheros *.mp3*. Para el uso de la playlist en todo la aplicación en la esquina superior izquierda aparece unos pequeños botones para reproducir, pausar, adelantar o atrasar las canciones. El comportamiente será el loop así nunca tendremos que preocuparnos si vamos a llegar al final de la playlist.

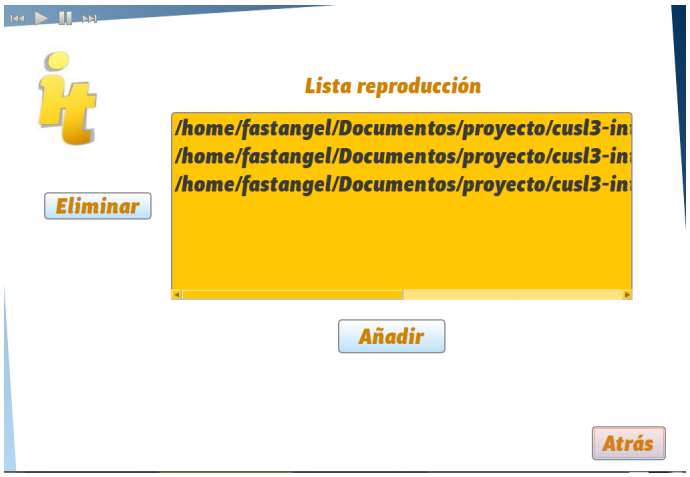

Figura 10.10: Playlist

10.12. Estadísticas

En la siguiente imagen 10.11 encontramos las estadísticas y la evolución del usuario. Si tenemos sufuciones datos como para tener una evolución nos aparecerá una nueva pestaña para elegir si mostramos los gráficos en tarta o un gráfico con la evolución.

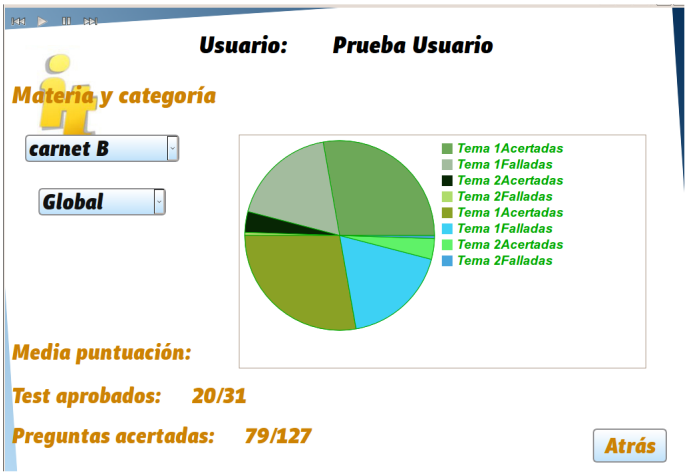

Figura 10.11: Estadísticas

10.13. Realización del test

En la siguiente imagen 10.12 encontramos la realización de una partida. En esta pantalla en la esquina superior derecha obtenemos el tiempo que se tiene para realizar el test. En la parte superior central encontramos la imagen asociada a la pregunta actual. Luego en la parte central tenemos información a acerca del test (respuestas sin contestar y la materia y categoría a la que pertenece el test) y tres botones dos para navegar por el test y uno último para finalizar el test. Por último obtenemos las opciones a la pregunta.

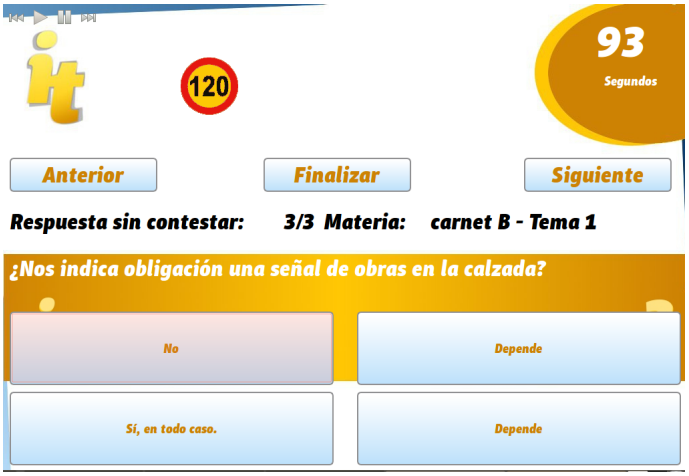

Figura 10.12: Realizar test

10.14. Resultados del test

En la siguiente imagen 10.13 encontramos los resultado obtenidos en el test. En esta pantalla únicamente se muestra datos referentes al test realizado y lo único que podemos hacer es volver al menú de usuario o ver el test realizado.

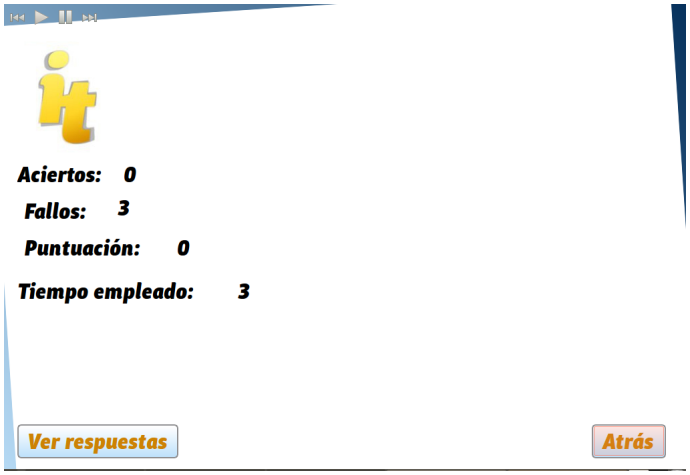

Figura 10.13: Resultado test

10.15. Ver test

En la siguiente imagen 10.14 encontramos la vista de visualizar un test finalizado. Nos aparecerá la pregunta, la respuesta nuestra y la correcta. Podremos navegar igual que si fuera el test real con la única excepción de que podemos irnos al menú de usuario cuando queramos.

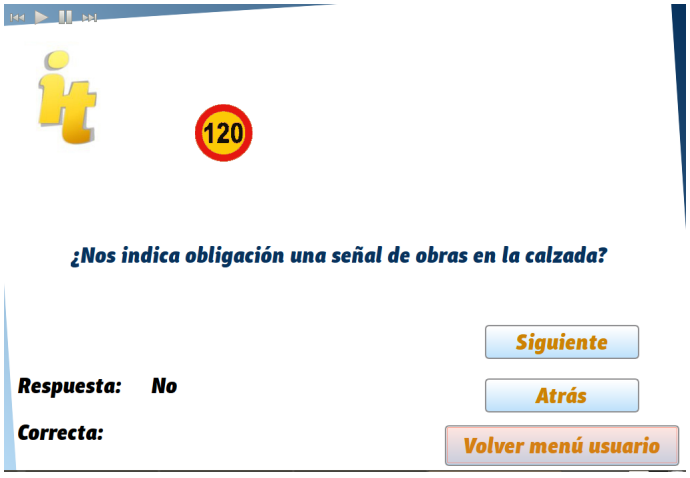

Figura 10.14: Resultado test

Capítulo 11

Manual de instalación

En siguiente manual es para la instalación y uso de la aplicación. Se dará por entendido que nos encontramos en en un sistema unix y que tenemos instaladas las librerías Qt y las extensiones qt-phonon, qmake, qt-sqlite y las utilidades para compilación de código C++ (g++ y make). La es compatible con la versión 4 de Qt.
La aplicación es distribuida a través de un fichero comprimido *tar.gz*. Lo primero que debemos hacer es bajarmos dicho fichero y colocarlos en la ubicación donde queremos instalarlo. Una vez que lo coloquemos en el lugar deseado abrimos una shell y nos movemos al directorio. Luego debemos ejecutar:

```
tar xvzf aplicacion.tar.gz
```

Una vez descomprimido nos movemos dentro de la carpeta que se ha creado. Se ha de prestar atención a que la carpeta img_preguntas tenga permisos de escritura para los usuarios que vayan a usar la aplicación:

```
cd aplicacion
```

Antes de poder comenzar a compilar la aplicación necesitamos instalar las librería de las que vamos a hacer uso. Si tienes un sistema unix ubuntu o debian nos bastará que buscar con synaptic los siguientes paquetes e instalarlos (*qt4-designer, qt4-dev-tools, qt4-qmake, qt4-qtconfig, libqt4-sql-sqlite, g++, libqt4-phonon*). En caso contrario de no tener un sistema unix ubuntu o debian deberemos acceder a la web oficial de cada librería y mirar la documentación de como se instala en el sistema que estemos utilizando.
Una vez dentro e instalado las dependencias debemos ejecutar la aplicación qmake para generar el makefile correspondiente a la aplicación:

```
qmake
```

Ahora debemos comprobar que se ha generado un fichero makefile. Si es así el siguiente paso es compilar la aplicación:

```
make
```

Si nos da un aviso o error de que falta alguna aplicación por instalar debemos instalarla antes de continuar. Una vez compilada la aplicación debemos instalar las fuentes para la aplicación. Esto consistirá en copiar todo lo que se encuentre dentro del directorio font de la aplicación al directorio del sistema */usr/share/fonts*. En caso de encontrarse en otro lugar las fuentes deberá copiarse a dicho lugar. Si no tenemos permiso para copiarla con el usuario actual deberemos copiarla con otro usuario que si la tenga o pueda realizar **sudo**:

```
cp /font/ChinoITCPro-BlackItalic.tff /usr/share/fonts
```

En caso de no obtener error al compilar ya tenemos la aplicación lista par ser ejecutada:

```
cd bin && .informatica_training
```

Capítulo 12

Manual del desarrollador

12.1. Introducción

En este capítulo comentaré algunos consejos para futuros desarrolladores que puedan hacer uso de la aplicación para desarrollar nuevas funcionalidades o cambios en la implementación.

12.2. Organización de carpeta

La organización de las carpetas es la siguiente:

- **BD:** En este directorio se almacena la BD. El nombre de está es *BD.sqlite*
- **bin:** En este directorio se almacena el binario de la aplicación con el nombre *informatica_training*
- **font:** En este directorio se encuentra la fuente utilizada para los formularios. Está exportada a ttf para poder instalarse en el sistema además se adjunta la original en otf y la aplicación para convertirla. Esta fuente es privativa por lo tanto si se instalar se debe aceptar su licencia.
- **forms:** En este directorio se encuentra todos los formularios en formato xml en una de las secciones posteriores explicaré como poder editarlos y crear alguno nuevo.
- **img:** En este directorio se almacenan todas las imágenes de las que hace uso la aplicación.
- **img_preguntas:** En este directorio se irán almacenando todas las imágenes que se asocien con las preguntas.
- **includes:** En este directorio se almacenarán todos los ficheros que no generen código ejecutable principalmente ficheros *.h*
- **mockups:** En este directorio se incluyen diseños de las interfaces. Suelen ser esbozos que luego puedan ayudar al diseñador de las interfaces.
- **music:** En este directorio se introducirán algunas músicas tranquilas con licencia libre para poder comenzar la playlist. No es necesario que las canciones vayan en este carpeta. Se pueden añadir canciones que se encuentren en cualquier parte del sistema.
- **objects:** Directorio utilizado para la compilación de la aplicación.
- **src:** Directorio donde se encuentra las archivos que generan código ejecutable. Principalmente ficheros *.cpp*

- **translates:** Aquí introduciremos los ficheros de traducción a usar por la aplicación.
- **xml_sample:** En este directorio se encuentra algunos ejemplos de xml para se utilización en la aplicación.

12.3. Modifcación de la Base de datos

Para la realización de modifaciones a la base datos recomiendo usar un plugin para el navegador web firefox llamado **SQLite Manager** y que tiene el siguiente aspecto:

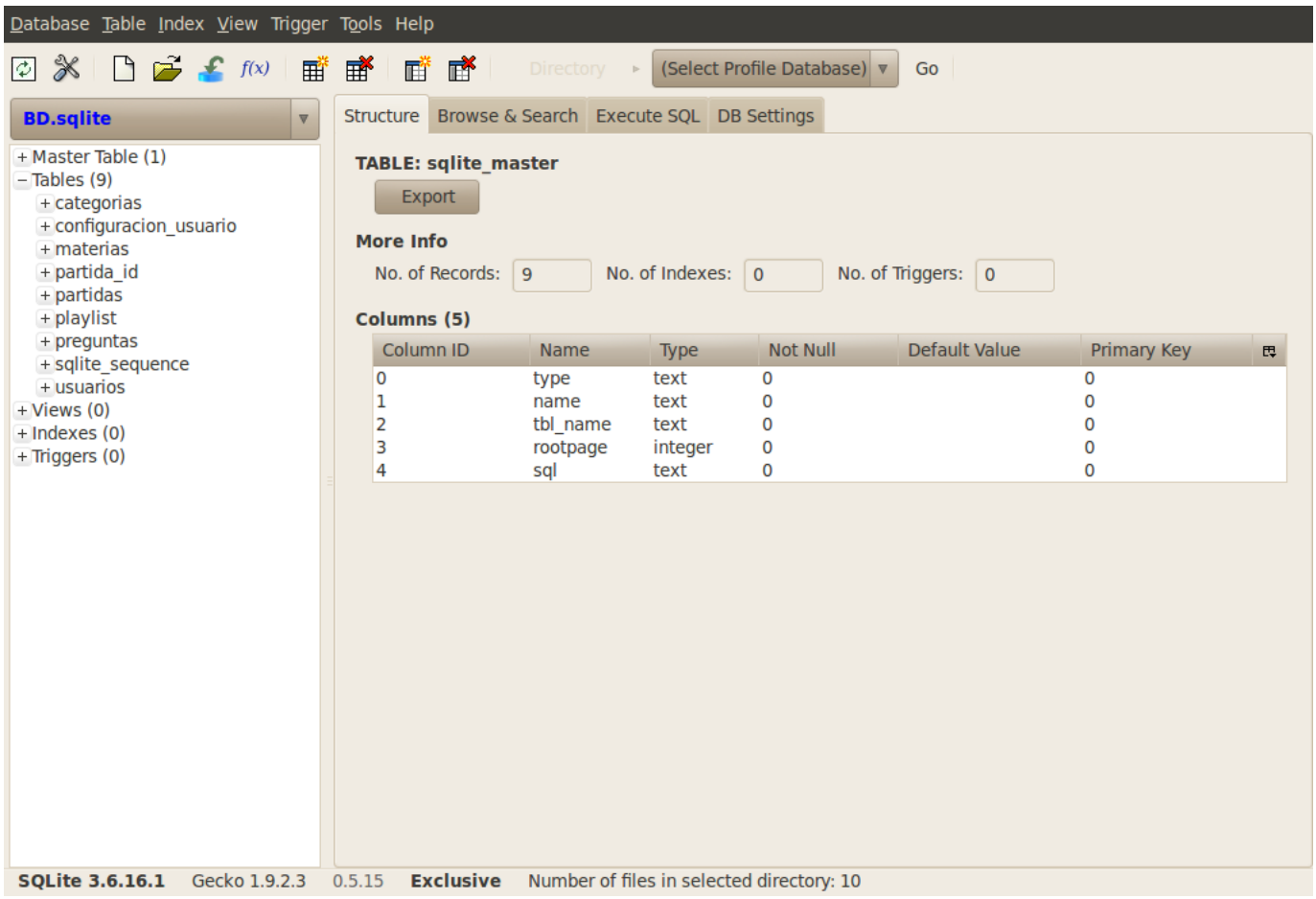

Figura 12.1: SQLite Manager

Como se puede observar la aplicación nos permite la realización de varias tareas. Las funciones que podemos realizar con la aplicación son las siguientes:

- Creación de tablas nuevas.
- Consulta a la BD.
- Consulta de los valores.
- Consulta de la estructura de la tablas.

- Modificación y edición de la estructuras de las tablas.
- Insercción, modifcación y elminación de filas de tablas.
- Configuración y optimización de la base de datos.
- Definir funciones por el usuario.

12.4. Modificación y creación de nuevos formularios

Para la creación y modifación de formularios se hace uso de una herramienta que proporciona Qt llamada **Qt Designer**. La usada para este proyecto es la versión 4. En la siguiente imagen 12.2 podemos observar el aspecto visual de esta.

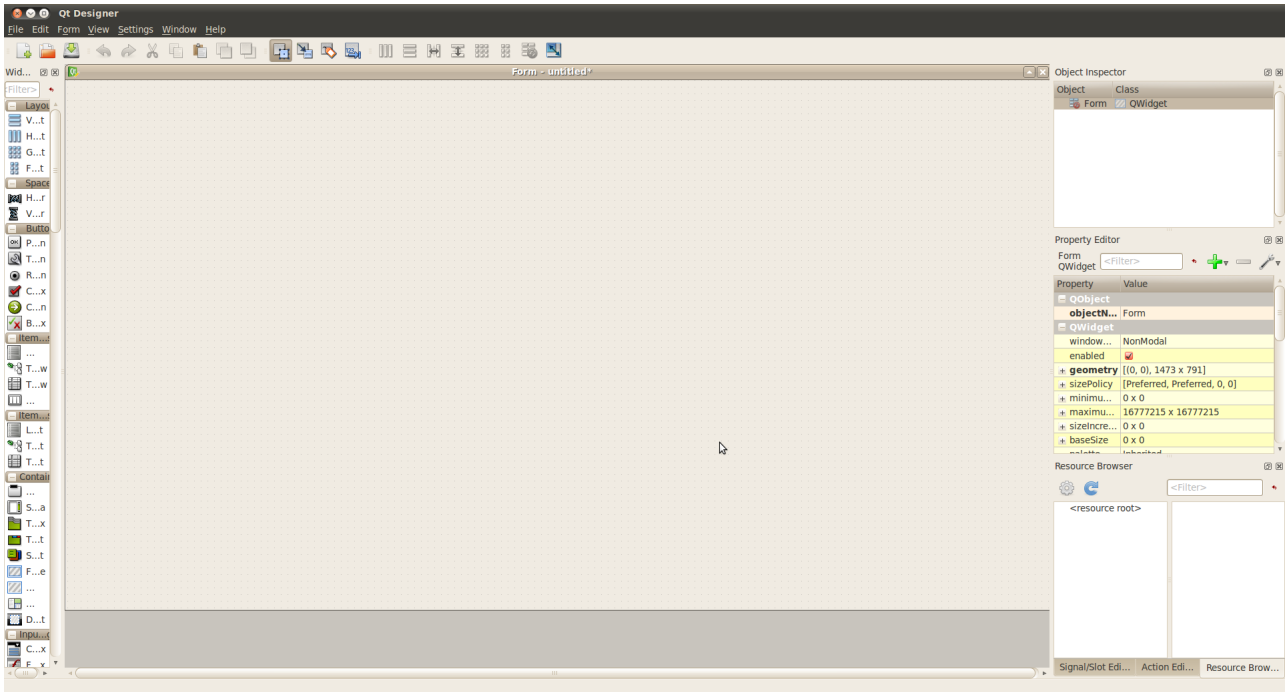

Figura 12.2: Qt4 Designer

Al entrar en la aplicación se nos mostrará unas seleccionar para el tipo de formulario que vamos a crear 12.3 yo recomiendo implementar siempre los formularios con el tipo widget por que es algo muy genérico y con lo que podemos llegar de una forma sencilla a conseguir resultados muy buenos. Si se está buscando un tipo de formulario específico y que sabemos que existe en Qt entonces si es mejor usar ese otro tipo de formulario.

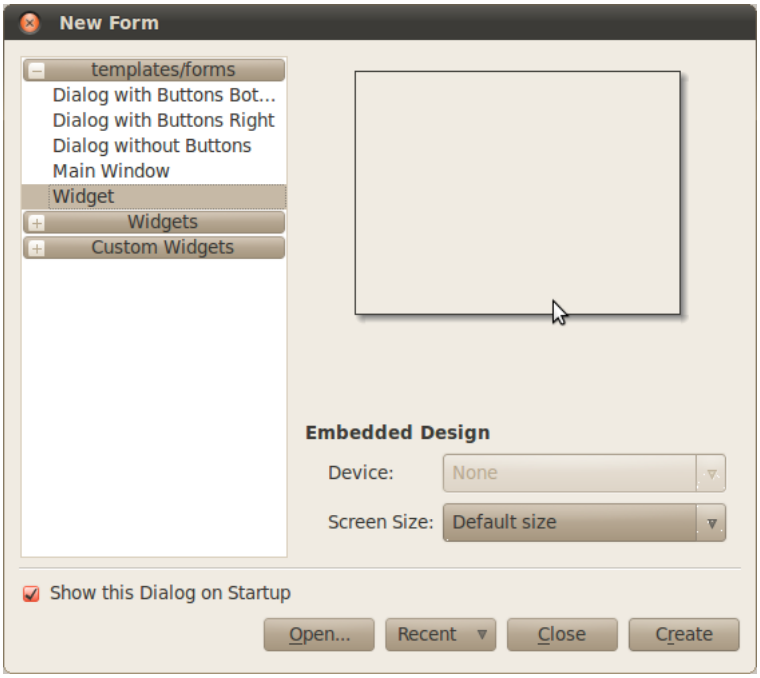

Figura 12.3: Qt4 Designer entrada

Una vez elegido el tipo de widget que queremos crear pasamos a comentar las distintas regiones en que se divido Qt Designer. En la siguiente imagen 12.4 encontramos que hemos dividido en 6 partes principales la aplicación comenzaré por orden a explicarlas.

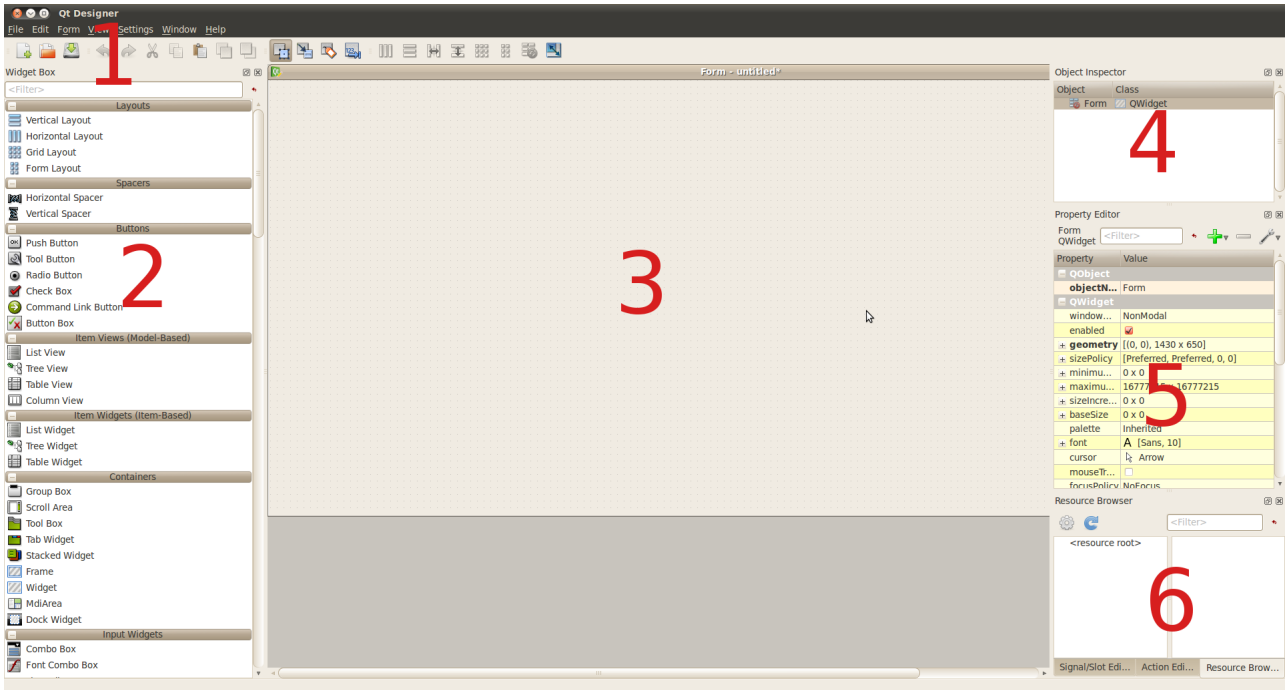

Figura 12.4: Qt4 Designer regiones

12.4.1. Región 1

En esta primera región encontraremos una barra de botones. Los tres primeros botones son fáciles de entender crear un nuevo formulario, abrir uno existente y guardar el que estemos editando actualmente. Luego nos encontramos varios botones que no son de especial interés exceptuando tres:

- **Botón editar widgets:** Como se puede observar en la imagen 12.5 con este botón activamos la vista de edición de widgets. Pudiendo añadir nuevos widgets o modificar los existenes en la rejilla del formulario.

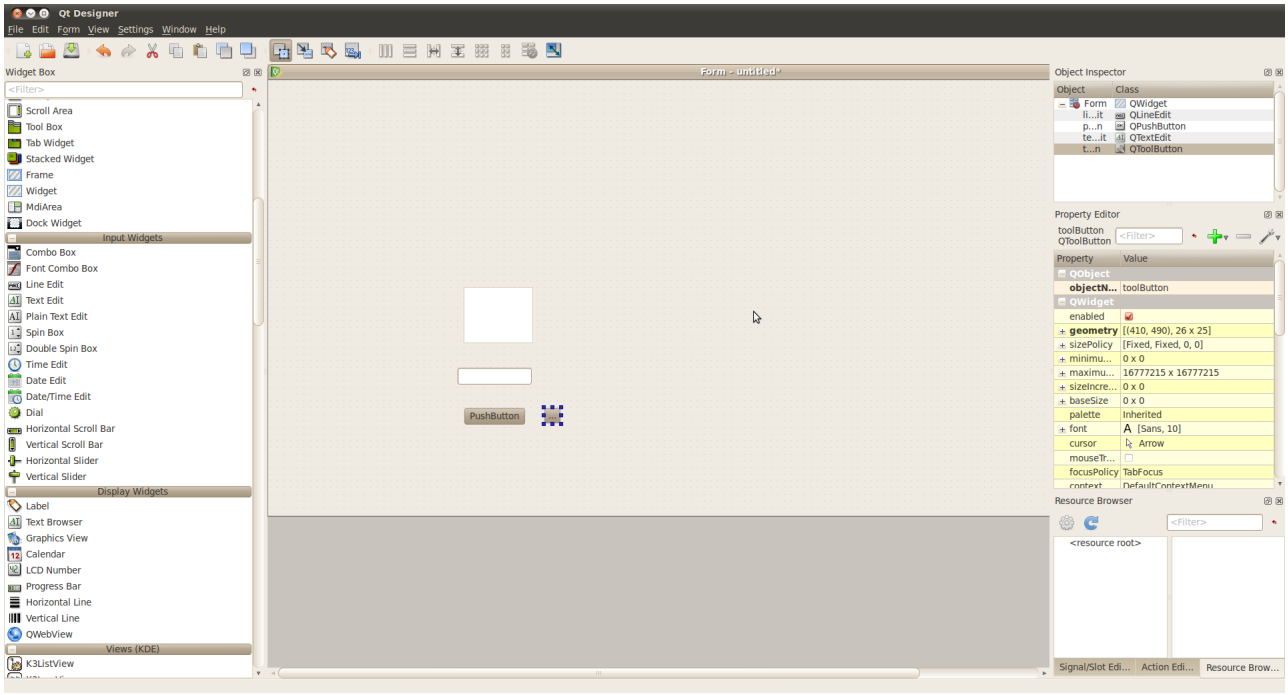

Figura 12.5: Qt4 Designer editar widgets

- **Editar señales y slots:** Como se puede observar en la imagen 13.13 con este botón activamos la vista de editar señales y slots. En ella podremos configurar que señales reciben los widgets y que slots se ejecutará en consecuencia.

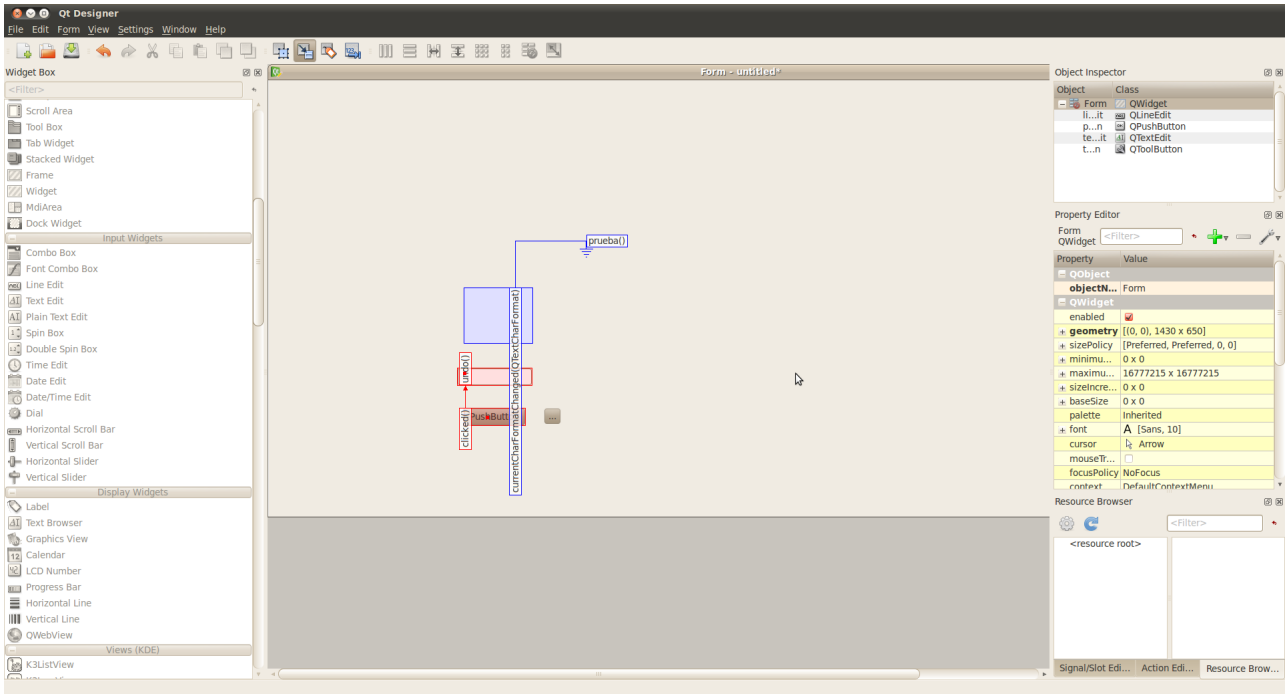

Figura 12.6: Qt4 Designer editar señales y slots

- **Editar tab order:** Como se puede observar en la imagen 12.7 con este botón cambiamos el orden de tabulación. Al pulsar en un widget se irá cambiando el número de tab order y si seguimos pulsando se irá incrementando.

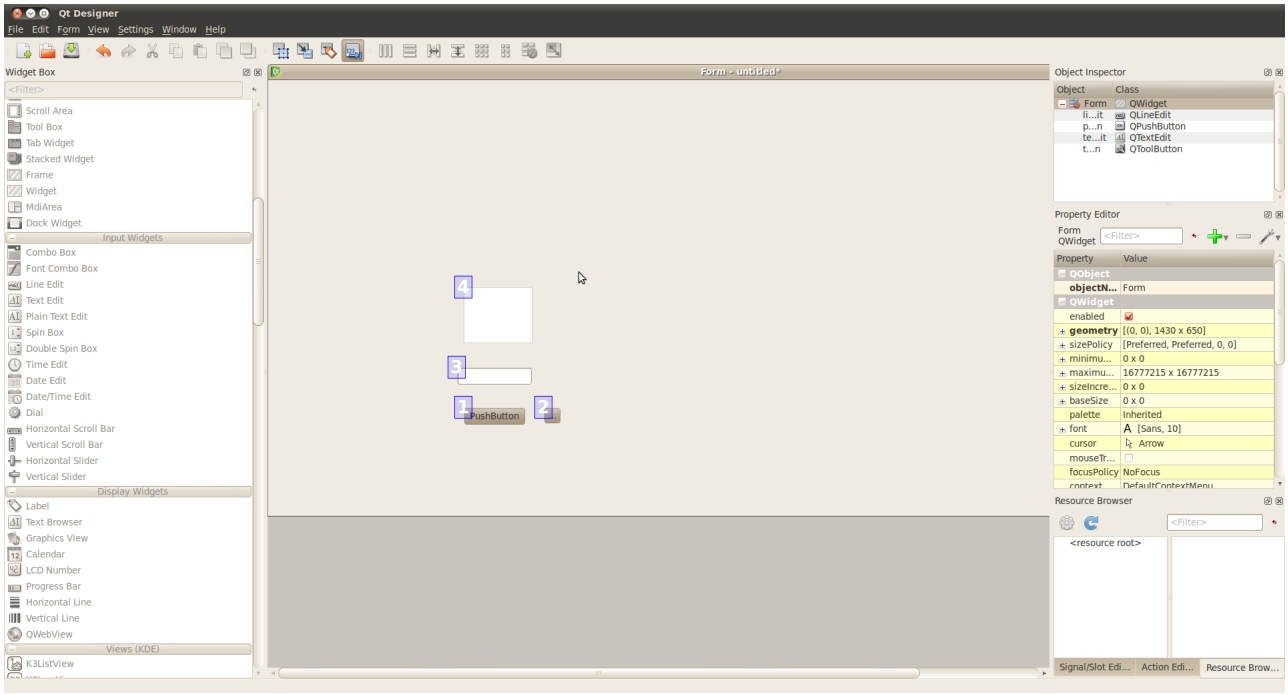

Figura 12.7: Qt4 Designer ordenación de tabulación

12.4.2. Región 2

En esta segunda región se muestra las distintos tipos de widget que se pueden añadir al formulario. También se pueden añadir nuevos widget creados por nosotros. Esta segunda región se divide en varias secciones:

- **Layouts:** Utilidades para crear layouts en los formularios.
- **Spacers:** Utilidades para añadir espaciadores en los formularios.
- **Buttons:** Diferences tipos de botones. El más usado es push Button aunque el radio button y el check box también son muy útiles.
- **Item Views:** Aquí encontramos los elementos típicos para ver varios elementos (listas, árboles, tablas, columnas).
- **Item Widgets:** Aquí encontramos lo mismo que lo anterior que basado en elementos en vez de basados en modelos.
- **Containers:** Contenedores para agrupar varios widgets. El que más suelo utilizar es Widget para poder introducir un widget dentro de otro.
- **Input Widgets:** Elementos para introducir datos los más utilizado son los combo y los text.
- **Display Widgets:** Elementos para mostrar datos de una forma visual en el formulario.
- **Views (KDE):** Igual que los anteriores pero implementados con las librerías de KDE y añadiendo nuevos elementos.
- **Containers (KDE):** Igual que los contenedores anteriores pero inmplementados con las librerías de KDE y añadiendo nuevos elementos.
- **Graphics (KDE):** Utilidades para usar gráficos pero con las librerías de KDE y añadiendo nuevos elementos.
- **Input (KDE):** Al igual que los Inputs anteriores pero usando las librerías de KDE y añadiendo nuevos elementos.
- **Buttons (KDE):** Al igual que los Buttons pero usando las librerías de KDE y añadiendo nuevos elementos.
- **Display (KDE):** Al igual que los Display pero usando las librerías de KDE y añadiendo nuevos elementos.
- **Plot (KDE):** Utilidades de KDE para dibujar gráficas.
- **Sonnet (KDE):** Utilidades de KDE para usar un diccionario.
- **Arthur Widgets**

  $$Demo$$

  : Ejemplos de implementación de widgets widgets.
- **Display Widgets**

  $$Examples$$

  : Ejemplos de implementación de widgets.
- **Phonon:** Utilidades de elementos multimedia (sonido, vídeo, ...).
- **Qt3 Support:** Widgets que se usaban en la versión anterior de Qt.

12.4.3. Región 3

En esta región es la región de diseño del formulario. Lo que se ponga aquí es lo que va a contener el formulario y sería como previsualización del formulario aunque no coincida luego exáctamente. Para previsualizarlo lo hacemos con **CTLR+R**.

12.4.4. Región 4

En esta región encontramos todos los widgets que hemos añadido al formulario y su jerarquía. Además desde aquí podremos cambiar el nombre de los widges para ponerle alguno que nos sea fácil de recordar a la hora de programar.

12.4.5. Región 5

En esta región tenemos la configuración del widget que estamos editando ahora mismo. Podemos cambiar los valores para que se ajuste a nuestras necesidades.

12.4.6. Región 6

Por último tenemos la región de las señales y slots en donde podremos editar los slots que hemos creado y las señales que le hemos asociado.

12.5. Internacionalización y generación de plantillas de traducción

En esta sección explicaré la forma de ralizar una aplicación multidioma en Qt. Lo primero que debemos tener en cuenta a la hora de programar son dos cuestiones:

- Si nos encontramos dentro de un método de una clase *QObject* tenemos la función *tr()* para poder traducir cadenas. Todas las cadenas que pasen por esta función serán traducidas el idioma actual del sistema.

- Si nos encontramos fuera de una clase *QObject* debemos realizar las traducciones de la siguiente forma *QApplication::translate("GestionMaterias", "Todas categorías", 0, QApplication::UnicodeUTF8)*. Esta función le debemos pasar primero el formulario al que pertecene la cadena, la cadena, parámetro de desambiguar y por último la codificación de la cadena a traducir.

Una vez finalizada la aplicación debemos ejecutar el siguiente comando para extraer en un fichero todas las cadenas a traducir de la aplicación (Las cadenas serán obtenidas a través del fichero *.pro* por lo que debemos ejecutar dicho comando en donde se encuentre este fichero):

```
lupdate informatica_training.pro
```

En el fichero *informatica_training.qm* tendremos todas las cadenas el siguiente paso es traducirlas para ello utilizamos la utilidad que nos proporciona Qt llamada **Qt Linguist**:

```
linguist informatica_training.qm
```

Y nos aparecerá una ventana como la siguiente donde deberemos configurar el idioma original de las cadenas y al idioma que van a ser traducidad.

Figura 12.8: Selección idioma en Qt Linguist

Una vez seleccionado el idioma original de las cadenas y el idioma al que lo vamos a traducir se nos abrirá la siguiente interfaz:

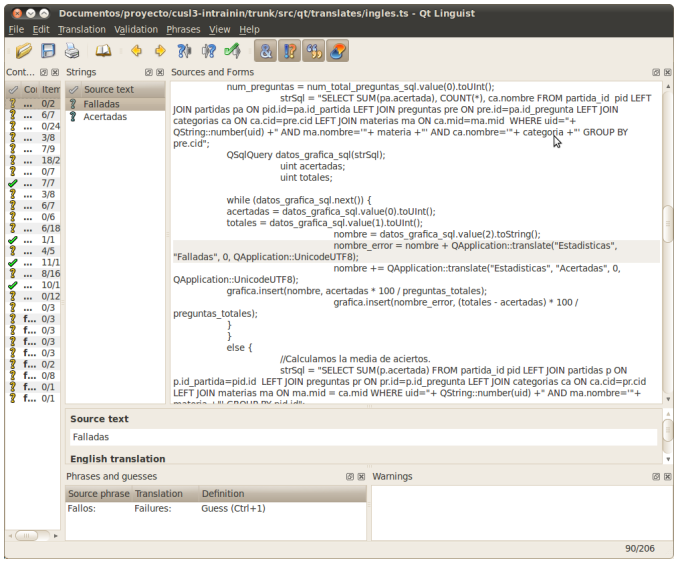

Figura 12.9: Traducción de cadenas en Qt Linguist

Esta interfaz la podemos dividir en las siguientes regiones:

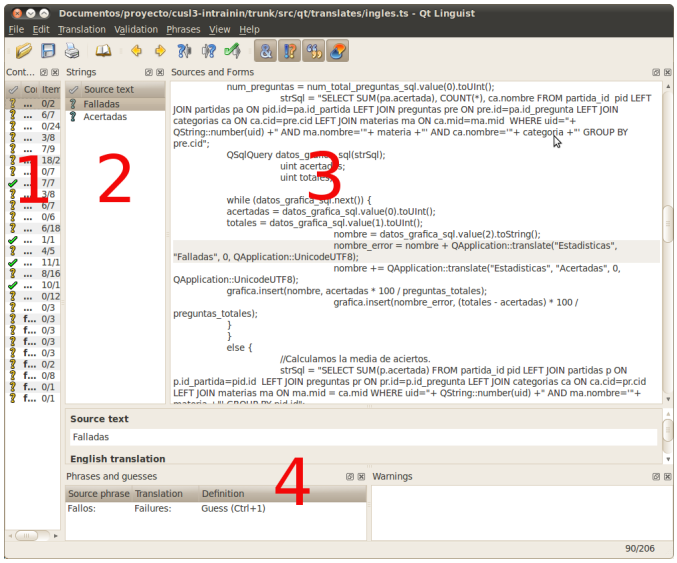

Figura 12.10: Regiones de Qt Linguist

A continuación describiré todas las regiones:

- **Región 1:** En esta región se nos muestra las cadenas que hay por ficheros y las que tenemos traducidas. Si no tenemos todas traducidas nos aparecerá una *?* en caso contrario nos aparecerá una *v*.
- **Región 2:** En esta región nos aparecen las cadenas originales del fichero que estamos traducción. En el ejemplo tendríamos dos cadenas *Falladas* y *Acertadas* sin traducir. A la izquierda de cada cadena sucede lo mismo que en la región 1. Si no está traducida aparecerá una *?* y en si lo está aparecerá una *v*.
- **Región 3:** En esta región aparece la situación en donde se encuentra la cadena a traducir. Mostrará el fichero de código donde la escribimos durante la implementación y nos la resaltará para poder estar seguro que es la cadena que queremos traducir.
- **Región 4:** En esta región es donde debemos colocar la cadena traducida. Nos aparece la fuente de la cadena y deberemos traducirla al idioma adecuado. En este ejemplo al inglés.

Capítulo 13

Introducción a Qt con Qt-designer

13.1. Introducción

En este apartado describiré como crear una pequeña aplicación Qt empleando para el diseño Qt-designer.

13.2. Diseño del formulario

El formulario que vamos a crear es un formulario sencillo. Trendrá tres widgets un botón, un input y un label.
El funcionamiento será el siguiente al introducir un valor en el input y pulsar el botón el texto introducido aparecerá en el label. Pero si el valor introducido es manzana se la aplicará un estilo rojo al texto, si el valor introducido es limón se le aplicará un estilo amarillo y en caso contrario se colocará un color negro.
El diseño de la interfaz queda de la siguiente forma:

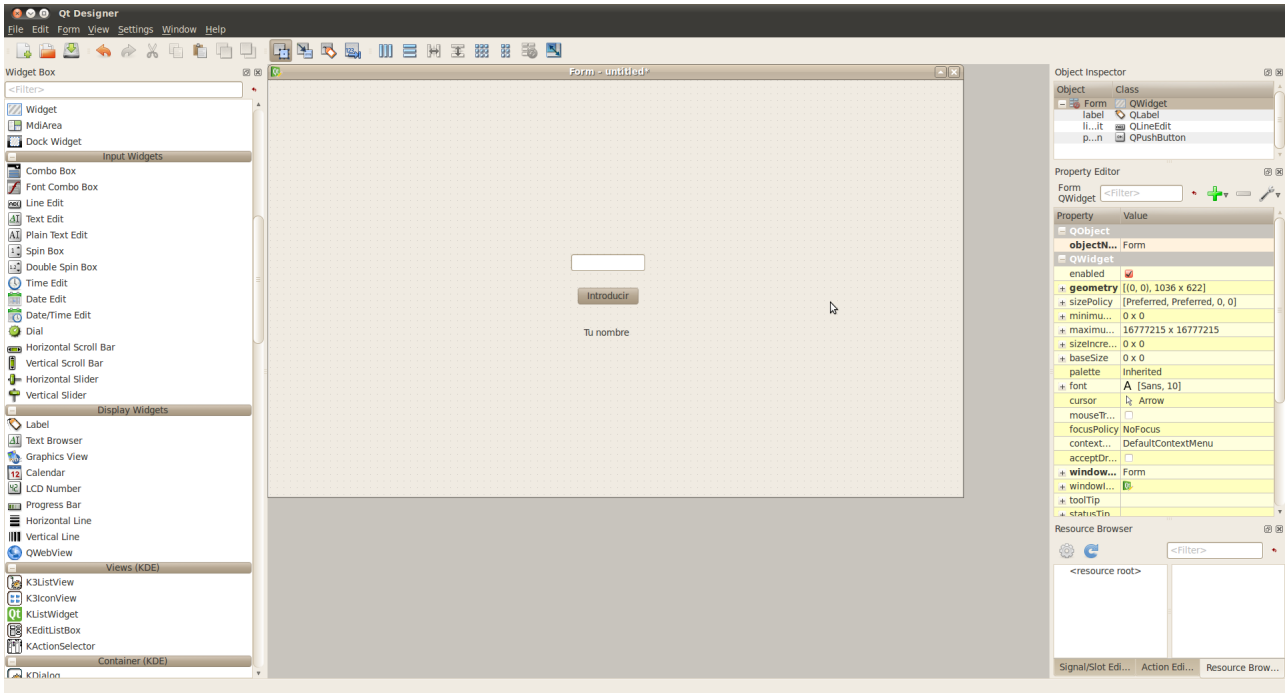

Figura 13.1: Diseño del widget de ejemplo

Una vez diseñada queda algo sosa por lo que podemos aplicar algún estilo CSS a cada objeto para que quedo mas vistoso. Por lo tanto tras aplicar unos estilos la interfaz queda de la siguiente forma:

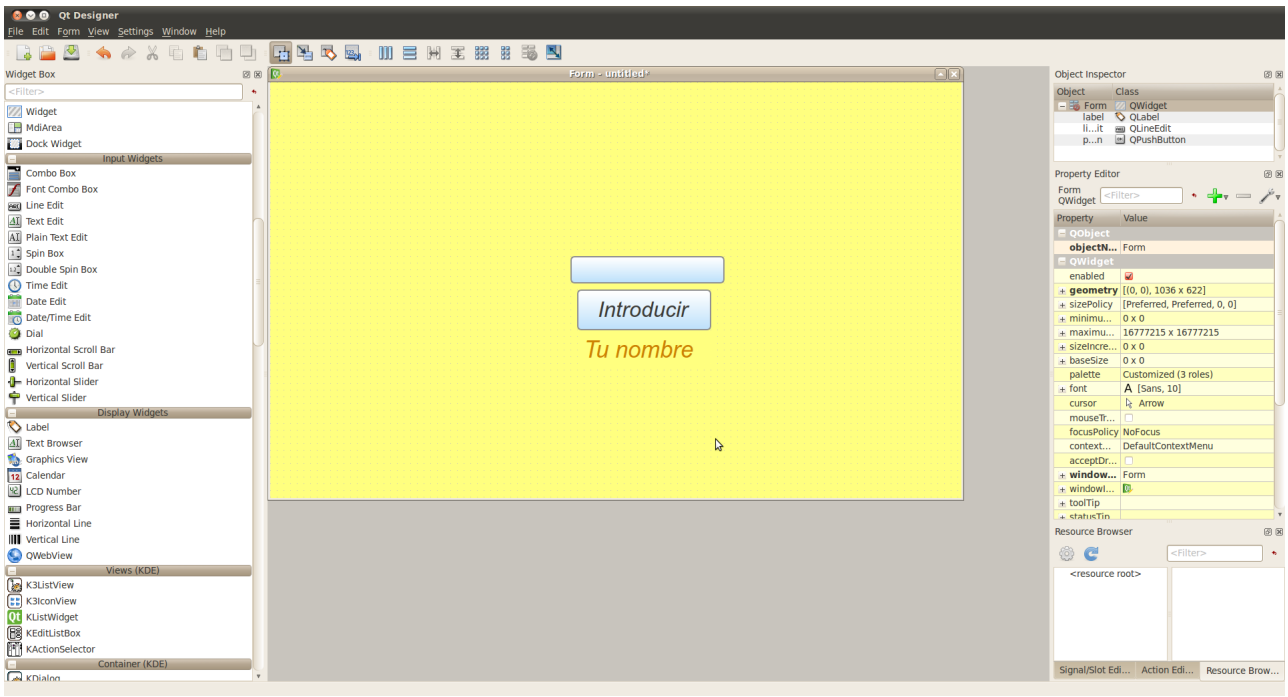

Figura 13.2: Diseño del widget de ejemplo con estilos

Los estilos CSS aplicado a cada widget son los siguientes:

- Formulario global:

```
background-color: rgb(255, 255, 127)
```

- lineEdit:

```
background-color: qlineargradient(spread:pad, x1:1,
 y1:1, x2:1, y2:0.0573182, stop:0
 rgba(189, 225, 251, 255), stop:1 rgba(255, 255, 255, 255));
font: 87 italic 24pt "Arial";
border: 2px solid #8f8f91;
border-radius: 6px;
```

- pushButton:

```
background-color: qlineargradient(spread:pad, x1:1,
 y1:1, x2:1, y2:0.0573182, stop:0
 rgba(189, 225, 251, 255), stop:1 rgba(255, 255, 255, 255));
font: 87 italic 24pt "Arial";
border: 2px solid #8f8f91;
border-radius: 6px;
```

- label:

```
        font: 87 italic 26pt "Arial";
    color: rgb(205, 130, 3);
```

13.3. Creación de los SLOTS

Una vez diseñado toca decicir el diseño de los *SLOTS*. Para ello nos vamos a la pestaña de las señales y slots:

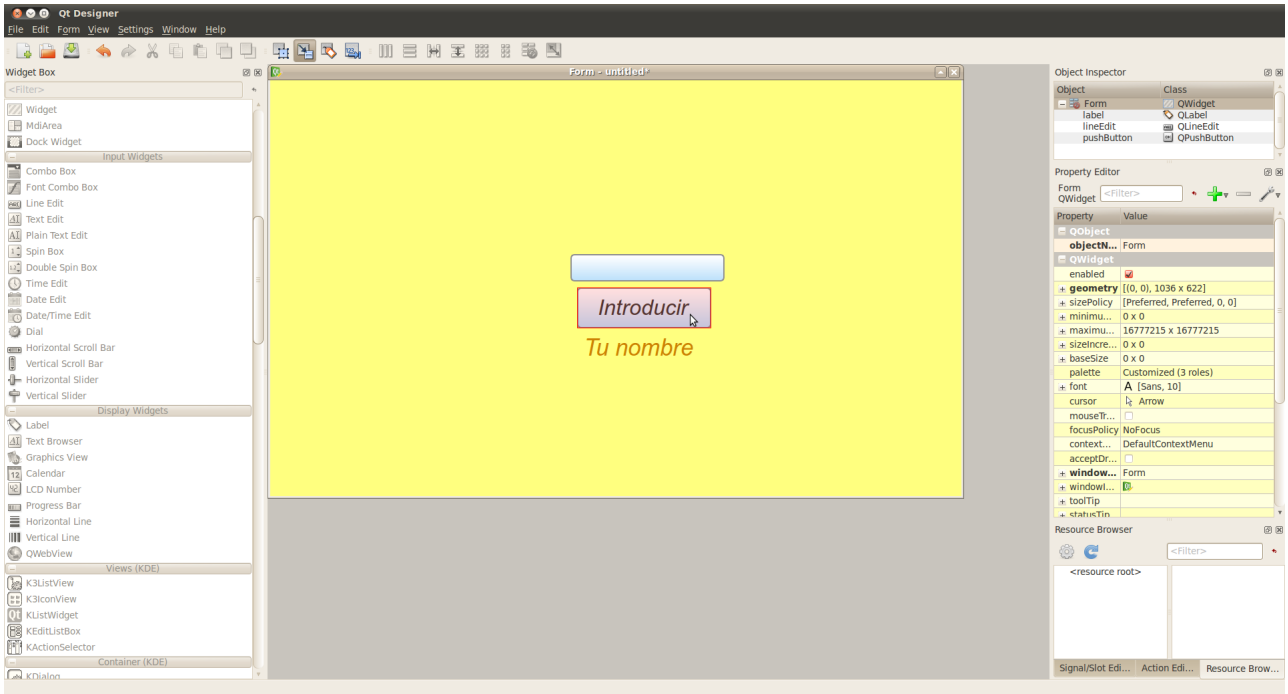

Figura 13.3: Diseño del widget de ejemplo señales y slots

Como podemos ver en la imagen nos aparece de nuevo el formulario pero con las señales asignadas. En este caso como no tenemos señales ni SLOTS no nos aparecerá ninguno.
El siguiente paso es crear un SLOT llamado *cambiar_nombre* asociado a la señal *click* del button. Para ello pulsamos en el button y sin soltar arrastramos hasta el formulario contenedor. Esto se hace de la siguiente forma:

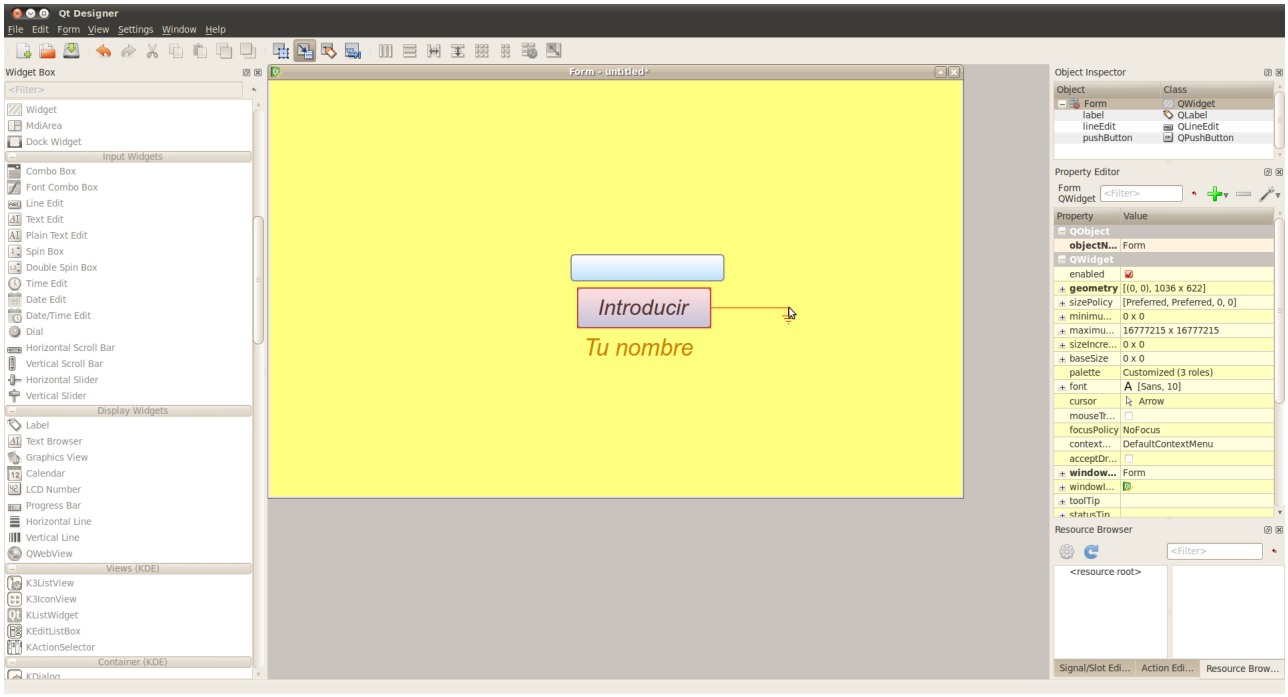

Figura 13.4: Diseño del widget de ejemplo crear SLOT

Al realizar esta acción se nos mostrará las siguientes opciones:

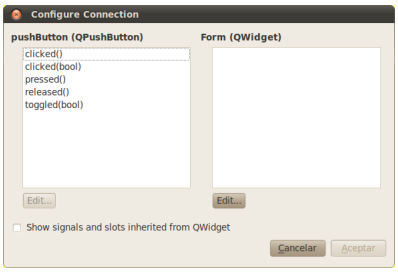

Figura 13.5: Diseño del widget de ejemplo crear nuevo SLOT

En la parte de la izquierda seleccionamos *clicked()* en la parte de la derecha añadimos nuestro SLOT llamado *cambiar_nombre*.
Nos deberá quedar de la siguiente forma:

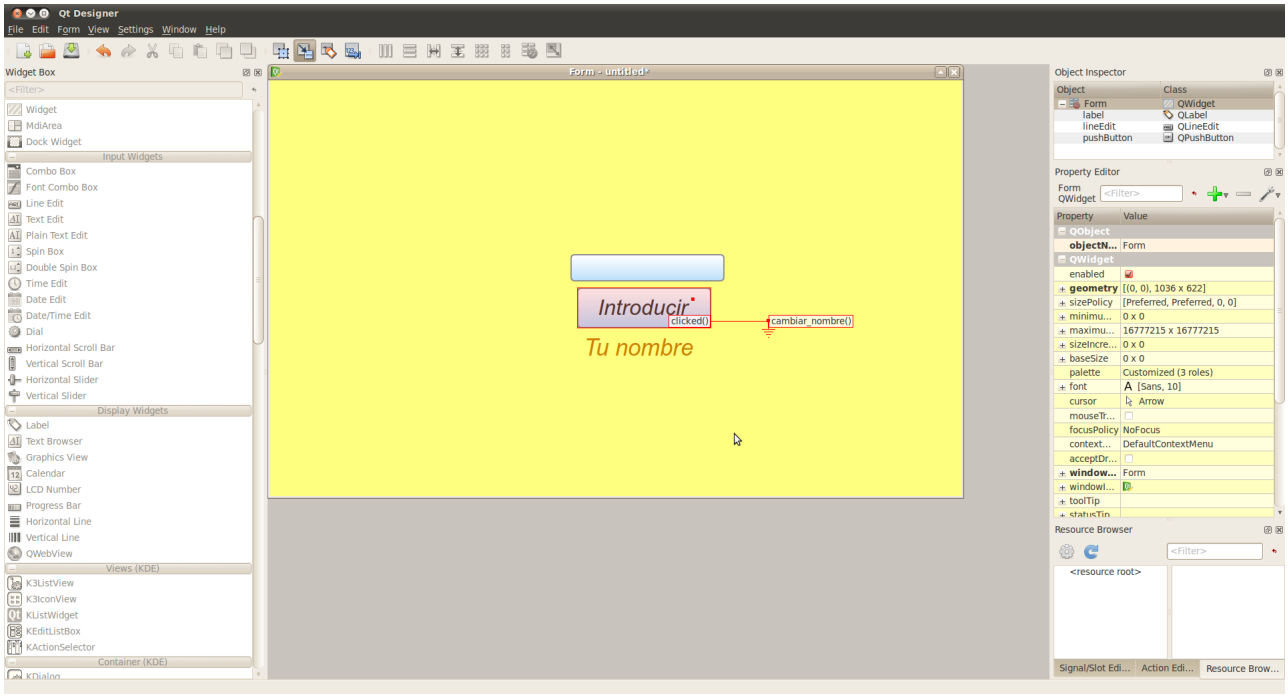

Figura 13.6: Diseño del widget de ejemplo muestra de SLOT

Ya solo nos queda guardar nuestro formulario recién creado. Para seguir concordancia con los datos se guardará con el nombre *ejemplo.ui*

13.4. Fichero configuración Qt

Ahora pasamos a crear el fichero de configuración de Qt. El fichero se llamará *ejemplo.pro* y estará en el mismo lugar donde guardamos el formulario. El fichero contendrá lo siguiente:

```
TEMPLATE = app
QT += core
CONFIG += uitools
FORMS += ejemplo.ui
HEADERS += ejemplo.h
SOURCES += main.cpp ejemplo.cpp
```

Las líneas que más nos importan a nosotros son *FORMS*, *SOURCES* y *HEADERS*. La primera debemos colocar nuestro formulario y en la segunda y tercera nuestro fichero de implementación del código.

13.5. Creación del formulario

Con el siguiente comando conseguiremos convertir el formulario que creamos anteriormente para su uso con C++.

```
/usr/bin/uic-qt4 ejemplo.ui -o ui_ejemplo.h
```

El resultado se llamará *ui_ejemplo.h* y contiene lo siguiente:

```
/*
    ********************************************************************************
** Form generated from reading UI file 'ejemplo.ui'
**
** Created: Sun May 16 09:03:23 2010
**      by: Qt User Interface Compiler version 4.6.2
**
** WARNING! All changes made in this file will be lost when recompiling UI
    file!
********************************************************************************
    */

#ifndef UI_EJEMPLO_H
#define UI_EJEMPLO_H

#include <QtCore/QVariant>
#include <QtGui/QAction>
#include <QtGui/QApplication>
#include <QtGui/QButtonGroup>
#include <QtGui/QHeaderView>
#include <QtGui/QLabel>
#include <QtGui/QLineEdit>
#include <QtGui/QPushButton>
#include <QtGui/QWidget>

QT_BEGIN_NAMESPACE

class Ui_Form
{
public:
    QPushButton *pushButton;
    QLineEdit *lineEdit;
    QLabel *label;

    void setupUi(QWidget *Form)
    {
        if (Form->objectName().isEmpty())
            Form->setObjectName(QString::fromUtf8("Form"));
        Form->resize(1036, 622);
        Form->setStyleSheet(QString::fromUtf8("background-color: rgb(255,
            255, 127)"));
        pushButton = new QPushButton(Form);
        pushButton->setObjectName(QString::fromUtf8("pushButton"));
        pushButton->setGeometry(QRect(460, 310, 201, 61));
        pushButton->setStyleSheet(QString::fromUtf8("  background-color:
            qlineargradient(spread:pad, x1:1, y1:1,
          x2:1, y2:0.0573182, stop:0 rgba(189, 225, 251, 255), stop:1 rgba
            (255, 255, 255, 255));\n"
"font: 87 italic 24pt \"Arial\";\n"
"border: 2px solid #8f8f91;\n"
"border-radius: 6px;"));
        lineEdit = new QLineEdit(Form);
        lineEdit->setObjectName(QString::fromUtf8("lineEdit"));
        lineEdit->setGeometry(QRect(450, 260, 231, 41));
```

```
49          lineEdit->setStyleSheet(QString::fromUtf8("  background-color:
                qlineargradient(spread:pad, x1:1, y1:1,
50           x2:1, y2:0.0573182, stop:0 rgba(189, 225, 251, 255), stop:1 rgba
                (255, 255, 255, 255));\n"
51  "font: 87 italic 24pt \"Arial\";\n"
52  "border: 2px solid #8f8f91;\n"
53  "border-radius: 6px;"));
54          label = new QLabel(Form);
55          label->setObjectName(QString::fromUtf8("label"));
56          label->setGeometry(QRect(470, 380, 191, 41));
57          label->setStyleSheet(QString::fromUtf8("font: 87 italic 26pt \"
                Arial\";\n"
58  "color: rgb(205, 130, 3);"));
59
60          retranslateUi(Form);
61          QObject::connect(pushButton, SIGNAL(clicked()), Form, SLOT(
                cambiar_nombre()));
62
63          QMetaObject::connectSlotsByName(Form);
64      } // setupUi
65
66      void retranslateUi(QWidget *Form)
67      {
68          Form->setWindowTitle(QApplication::translate("Form", "Form", 0,
                QApplication::UnicodeUTF8));
69          pushButton->setText(QApplication::translate("Form", "Introducir",
                0, QApplication::UnicodeUTF8));
70          label->setText(QApplication::translate("Form", "Tu nombre", 0,
                QApplication::UnicodeUTF8));
71      } // retranslateUi
72
73  };
74
75  namespace Ui {
76      class Form: public Ui_Form {};
77  } // namespace Ui
78
79  QT_END_NAMESPACE
80
81  #endif // UI_EJEMPLO_H
```

Como podemos observar es la declaración completa del formulario y nostros no hemos tenido que realizarla gracias a Qt-designer.

13.6. Implementación del SLOT

Para poder implementar el SLOT primero debemos crear una clase que haga uso del formularioy que declare el SLOT. Siguiendo el ejemplo la clase quedaría de la siguiente forma:
ejemplo.h

```
1  #ifndef EJEMPLO_H
2  #define EJEMPLO_H
3
4  #include "../includes/ui_ejemplo.h"
```

```
5
6  class ejemplo : public QWidget {
7    Q_OBJECT
8
9    public:
10      ejemplo(QWidget *parent = 0);
11   private slots:
12     void cambiar_nombre();
13   private:
14     Ui::Form ui;
15  };
16
17  #endif
```

ejemplo.cpp

```
1  #include <QtUiTools>
2  #include <QtGui>
3  #include <QString>
4
5  #include "ejemplo.h"
6
7
8  ejemplo::ejemplo(QWidget *parent): QWidget(parent) {
9    ui.setupUi(this);
10 }
11
12
13 void ejemplo::cambiar_nombre() {
14   QString nombre = ui.lineEdit->text();
15   QString style;
16
17   if (nombre == QString("manzana")) {
18     style = "font: 87 italic 26pt \"Arial\"; color: rgb(205, 130, 3);";
19   }
20   else if (nombre == QString("limon")) {
21     style = "font: 87 italic 26pt \"Arial\"; color: rgb(205, 130, 3);";
22   }
23   else {
24     style = "font: 87 italic 26pt \"Arial\"; color: rgb(205, 130, 3);";
25   }
26
27   ui.abel->setStyleSheet(style);
28   ui.label->setText(nombre);
29 }
```

main.cpp

```
1  #include <QApplication>
2  #include <QtUiTools>
3  #include <QtGui>
4
5  #include  "ejemplo.h"
6
7  int main(int argc, char *argv[]) {
```

```
8     QApplication *app = new QApplication(argc, argv);
9     app->setApplicationName("Ejemplo");
10    app->setQuitOnLastWindowClosed(true);
11    ejemplo formulario;
12    formulario.show();
13    return app->exec();
14  }
```

13.7. Compilación

Una vez implementado todo solo falta compilarlo. Para ello ejecutamos lo siguiente:

```
qmake && make
```

13.8. Ejecución

Por último solo queda visualizar el widget implementado. Para ello lo realizamos de la siguiente manera:

```
./ejemplo
```

Y algunos ejemplo de los uso de la aplicación son:

Figura 13.7: Diseño del widget de ejemplo previsualización 1

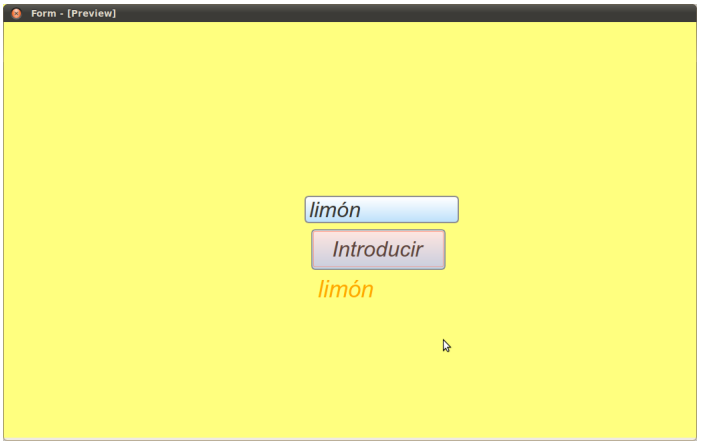

Figura 13.8: Diseño del widget de ejemplo previsualización 2

Figura 13.9: Diseño del widget de ejemplo previsualización 3

Software utilizado

En esta sección comentaré algunos programas que utilizado tanto para el desarrollo como para la documentación y creación de esta memoria.

Latex

LaTeX [5] es un sistema de composición de textos que está formado mayoritariamente por órdenes. La idea principal de LaTeX es que el autor se centra en el contenido y no en la forma del documento. Para lograr esto, LaTeX está provisto de una serie de macros y estilos predefinidos.

Doxygen

Doxygen [1] es utilizado para la documentación de código.
Esta herramienta realiza una documentación automática de código fuente. Es decir, para nuestro PFC, podemos utilizar para generar la documentación de las APIs de nuestras librerias y demás. Puede generar esta documentación en varios formatos, y entre ellos, LaTeX, de forma que podemos utilizar ese código generado en nuestra memoria de forma automática.

Qt 4

Qt 4 [3] [2] Qt es una biblioteca multiplataforma para desarrollar interfaces gráficas de usuario y también para el desarrollo de programas sin interfaz gráfica como herramientas de la consola y servidores.
Qt es utilizada principalmente en KDE, Google Earth, Skype, Qt Extended, Adobe Photoshop Album, VirtualBox y Opie.
Es producido por la división de software Qt de Nokia, que entró en vigor después de la adquisición por parte de Nokia de la empresa noruega Trolltech, el productor original de Qt, el 17 de junio de 2008.
Qt es utilizada en KDE, un entorno de escritorio para sistemas como GNU/Linux o FreeBSD, entre otros. Qt utiliza el lenguaje de programación C++ de forma nativa, adicionalmente puede ser utilizado en varios otros lenguajes de programación a través de bindings.

Funciona en todas las principales plataformas, y tiene un amplio apoyo. El API de la biblioteca cuenta con métodos para acceder a bases de datos mediante SQL, así como uso de XML, gestión de hilos, soporte de red, una API multiplataforma unificada para la manipulación de archivos y una multitud de otros para el manejo de ficheros, además de estructuras de datos tradicionales.

Distribuida bajo los términos de GNU Lesser General Public License (y otras), Qt es software libre y de código abierto.

Figura 13.10: Qt4 Designer editar señales y slots

Qt 4 Designer

Qt 4 Designer es una aplicación gráfica que nos proporciona Qt para la generación en XML de formularios. Una vez realizados estos formularios se pueden generar código que ya implemente el formulario completo para poder ser usado en una aplicación.

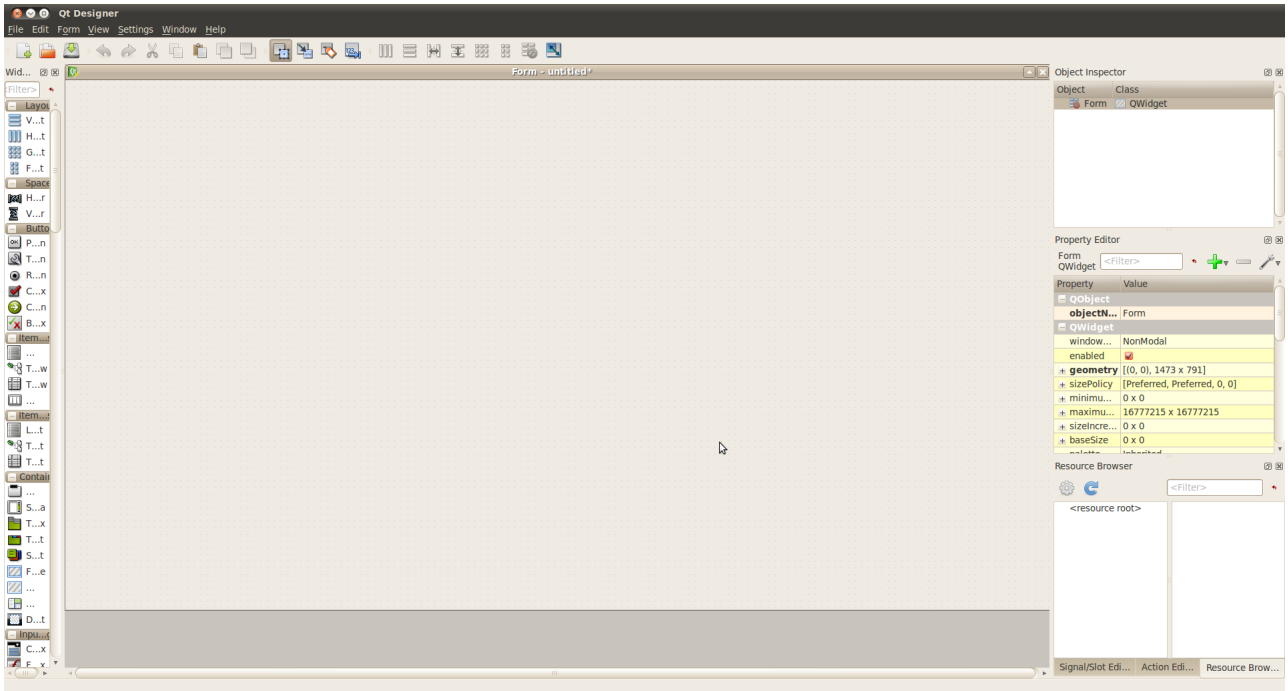

Figura 13.11: Qt4 Designer editar señales y slots

Qt 4 Linguist

Qt 4 Linguist es una aplicación gráfica que nos proporciona Qt para la traducción de aplicaciones. Es muy completa y nos permite traducir a varios idiomas así como compilar los ficheros de traducciones

para su posterior uso. Trae un sistema avanzada de traducciones para tener un gran control sobre las palabras.

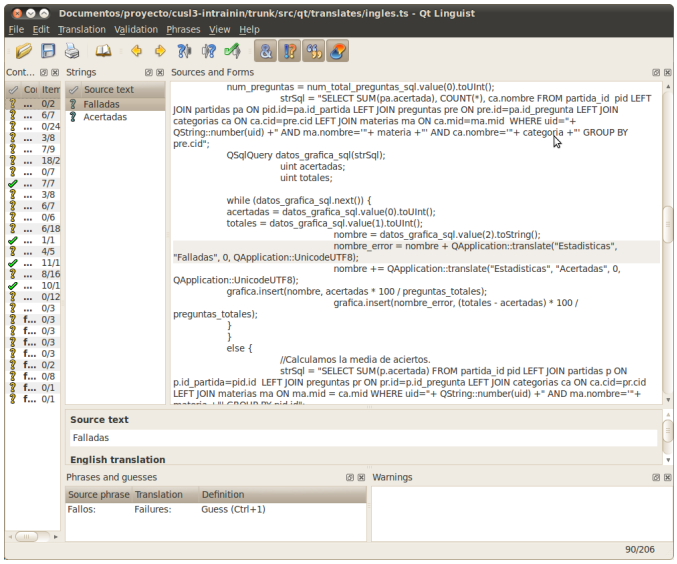

Figura 13.12: Qt4 Designer editar señales y slots

Qmake

Qmake es una utilidad que nos proporciona Qt para no tener que crear makefiles. Para ello creamos un fichero de configuración con la extensión textitpro y luego ejecutamos qmake y se encargará de buscar las dependencias y crear un makefile completo para poder compilar nuestra aplicación.

Phonon

Phonon Phonon es el framework multimedia estándar de KDE 4, también parte de Qt. [oo]
El objetivo de Phonon es facilitar a los programadores el uso de tecnologías multimedia en sus programas, así como asegurar que las aplicaciones que usen Phonon funcionen en diversas plataformas y arquitecturas de sonido.
Phonon crea una capa intermedia entre los programas de KDE y los diferentes motores multimedia. Esto permite a los programas usar el API estable de Phonon, independientemente de los cambios que se hagan en dichos motores. Un cambio en un motor multimedia sólo requerirá adaptar Phonon a dicho cambio, en vez de tener que adaptar cada una de las aplicaciones que usen dicho motor.

OmniGraffle Pro

OmniGraffle Pro es una aplicación del estilo de Día pero más avanzada. Con ella podemos crear diagramas muy complejos de una forma muy fácil. Trae infinidad de extensiones y el resultados de sus diagramas es muy profesional. Se ha utilizado para algunos diagramas para que queden más bonitos visualmente que **Día**.

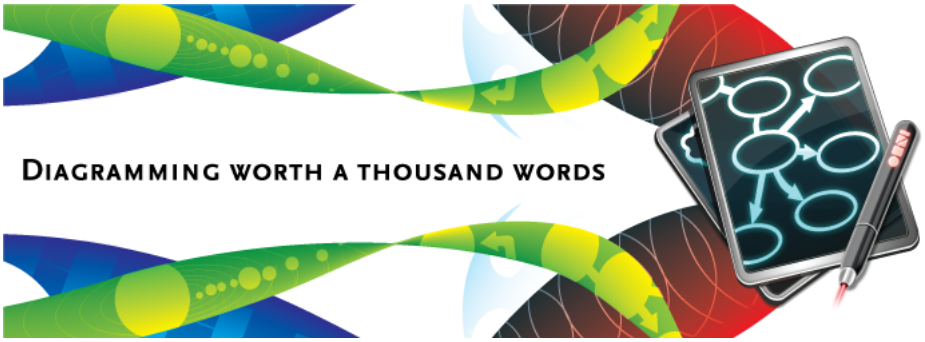

Figura 13.13: OmniGraffle Pro

Subversion

Subversion es un programa de control de versiones. Yo principalmente lo he usado como backups para tener registrado todos los cambios que he hizo realizando y si me confundía en algo restauraba. Además al usar esta aplicación es muy fácil trabajar en distintos ordenadores por que únicamente te bajas el repositorio modificas un fichero y lo vuelves a subir.
Una característica importante de Subversion es que, a diferencia de CVS, los archivos versionados no tienen cada uno un número de revisión independiente. En cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en un instante determinado. Se puede conseguir en [4]
Subversion puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintos ordenadores. A cierto nivel, la posibilidad de que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración. Se puede progresar más rápidamente sin un único conducto por el cual deban pasar todas las modificaciones. Y puesto que el trabajo se encuentra bajo el control de versiones, no hay razón para temer por que la calidad del mismo vaya a verse afectada —si se ha hecho un cambio incorrecto a los datos, simplemente deshaga ese cambio

Vim

Figura 13.14: Vim

Dia

Dia es un editor de gráficos vectoriales el cual incluye distintas plantillas para distintos tipos de gráficos, como pueden ser UML, ERe, diagramas de flujo, esquemas Cisco de red y un larguísimo etcétera. Podemos ver el interfaz en la figura 13.15

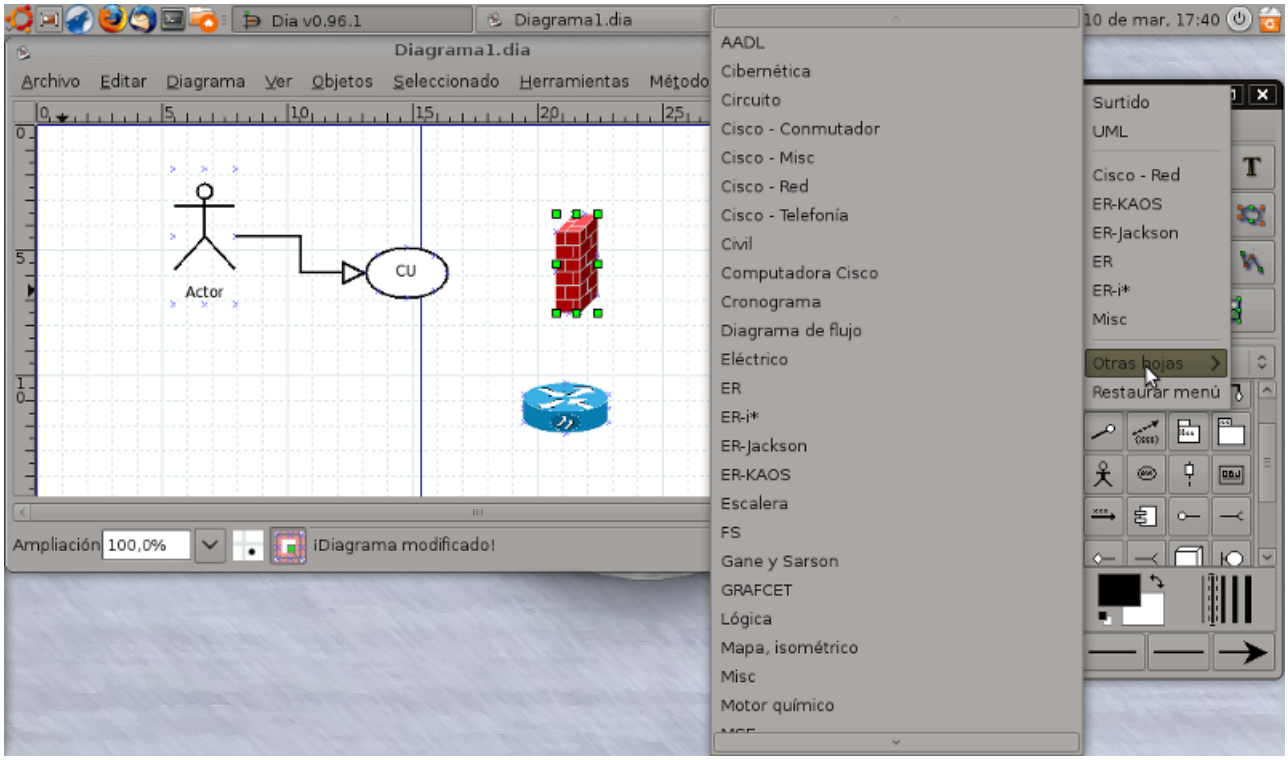

Figura 13.15: Interfaz de Dia

Estos diagramas podemos exportarlos a diversos formatos de imagen (`.png`, `.eps`, ...) o a formato `.tex`.

Bibliografía

[1] Dimitri Van Heesch. Pagina oficial de Doxygen. http://www.doxygen.org.

[2] Matthias Kalle Dalheimer. Programming with Qt, Second Edition. http://oreilly.com/catalog/9780596000646/index.html.

[3] Nokia Trolltech. Documentación de las API de Qt. http://doc.qt.nokia.com/.

[4] Subversion development team. Pagina oficial de subversion. http://subversion.apache.org.

[5] Wikibooks. The Book of LaTeX. http://en.wikibooks.org/wiki/LaTeX.

Instalación de Informática Training

En siguiente manual es para la instalación y uso de la aplicación. Se dará por entendido que nos encontramos en en un sistema unix y que tenemos instaladas las librerías Qt y las extensiones qt-phonon, qmake, qt-sqlite y las utilidades para compilación de código C++ (g++ y make). La es compatible con la versión 4 de Qt.
La aplicación es distribuida a través de un fichero comprimido *tar.gz*. Lo primero que debemos hacer es bajarmos dicho fichero y colocarlos en la ubicación donde queremos instalarlo. Una vez que lo coloquemos en el lugar deseado abrimos una shell y nos movemos al directorio. Luego debemos ejecutar:

```
tar xvzf aplicacion.tar.gz
```

Una vez descomprimido nos movemos dentro de la carpeta que se ha creado. Se ha de prestar atención a que la carpeta img_preguntas tenga permisos de escritura para los usuarios que vayan a usar la aplicación:

```
cd aplicacion
```

Antes de poder comenzar a compilar la aplicación necesitamos instalar las librería de las que vamos a hacer uso. Si tienes un sistema unix ubuntu o debian nos bastará que buscar con synaptic los siguientes paquetes e instalarlos (*qt4-designer, qt4-dev-tools, qt4-qmake, qt4-qtconfig, libqt4-sql-sqlite, g++, libqt4-phonon*). En caso contrario de no tener un sistema unix ubuntu o debian deberemos acceder a la web oficial de cada librería y mirar la documentación de como se instala en el sistema que estemos utilizando.
Una vez dentro e instalado las dependencias debemos ejecutar la aplicación qmake para generar el makefile correspondiente a la aplicación:

```
qmake
```

Ahora debemos comprobar que se ha generado un fichero makefile. Si es así el siguiente paso es compilar la aplicación:

```
make
```

Si nos da un aviso o error de que falta alguna aplicación por instalar debemos instalarla antes de continuar. Una vez compilada la aplicación debemos instalar las fuentes para la aplicación. Esto consistirá en copiar todo lo que se encuentre dentro del directorio font de la aplicación al directorio del sistema */usr/share/fonts*. En caso de encontrarse en otro lugar las fuentes deberá copiarse a dicho lugar. Si no tenemos permiso para copiarla con el usuario actual deberemos copiarla con otro usuario que si la tenga o pueda realizar **sudo**:

```
cp /font/ChinoITCPro-BlackItalic.tff /usr/share/fonts
```

En caso de no obtener error al compilar ya tenemos la aplicación lista par ser ejecutada:

```
cd bin && .informatica_training
```

GNU Free Documentation License

Version 1.3, 3 November 2008

<http://fsf.org/>

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.
This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.
We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "**Document**", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "**you**". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.
A "**Modified Version**" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.
A "**Secondary Section**" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "**Invariant Sections**" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.
The "**Cover Texts**" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.
A "**Transparent**" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "**Opaque**".
Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.
The "**Title Page**" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.
The "**publisher**" means any person or entity that distributes copies of the Document to the public.
A section "**Entitled XYZ**" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "**Acknowledgements**", "**Dedications**", "**Endorsements**", or "**History**".) To "**Preserve the Title**" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.
The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.
You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.
If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.
If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.
It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled “History”, Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled “History” in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.
You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.
You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.
The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.
In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.
You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.
If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.
If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.
However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.
Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.
Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.
Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.
"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.
"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.
An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.
The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with ... Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.
If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.